



Cole, Elizabeth (2023) *Considering primary school programming education using a comprehension-first approach*. PhD thesis.

<http://theses.gla.ac.uk/83607/>

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Enlighten: Theses

<https://theses.gla.ac.uk/>
research-enlighten@glasgow.ac.uk

Considering primary school introductory programming using a comprehension-first oriented approach.

Elizabeth Cole

Submitted in fulfilment of the requirements for the Degree of Doctor of Philosophy

School of Computing Science

College of Science and Engineering

University of Glasgow



University
of Glasgow

January 2022

Contents

Contents	iii
Tables	xi
Figures	xiv
Acknowledgements	xv
Declaration	xvi
Chapter 1	1
1.1 Introduction	1
1.2 The thesis statement and research questions	3
Abstract	3
1.3 Quadrant summaries	4
1.3.1 Quadrant A: Is Computing Science Education of Value to All?	4
1.3.2 Quadrant B: Typical CS approaches in primary education?	5
1.3.3 Quadrant C: A comprehension-oriented approach led by CS experts	7
1.3.4 Quadrant D: A comprehension-first approach led by primary teachers?	8
Chapter 2	12
2.1 Quadrant A: Chapter 2	12
2.2 Background and related work	13
2.2.1 Computing Science Education - A skill for All?	13
2.2.2 CS for "ALL" – The Journey	14
2.2.3 CS curricula collaborations with digital industries.	15
Chapter 3	18
The Study	18
3.1 Methods	18
3.1.1 Aims and objectives	18
3.1.2 Approach	18
3.1.3 Research Design	18

3.1.4	Setting	19
3.1.5	Participants	19
3.1.6	Materials.....	21
3.1.7	Procedure	22
3.1.8	Data Collection and Analysis.....	23
3.1.9	Soft skills.....	25
3.1.10	Hard skills	26
	Data processing and diagnostics	29
	Limitations	29
3.2	Results	30
3.2.1	Process related themes	31
3.3	Interview themes	32
3.3.1	Theme 1: Awareness of processes around them and the business problems that they solve	32
3.3.2	Theme 2: Information extraction and model building	34
3.3.3	Theme 3: Improvement via creation/adjustment of processes.....	36
3.3.4	Theme 4 Expressing processes verbally or in writing	39
3.4	Discussion	42
3.4.1	Process thinking and computer science.....	42
3.4.2	Theme 1: Awareness of processes around them and the business problems that they solve.	42
3.4.3	Theme 2 Processes for continuous improvement.....	43
3.4.4	Theme 3 Information extraction and knowledge representation	43
3.4.5	Theme 4 Process thinking for problem solving	43
3.5	Future Work	45
Chapter 4	48
4.1	Quadrant B	48
4.2	Introduction	48
4.3	Related work	50

4.4	Curriculum	51
4.5	Programming Tools.....	51
4.6	Materials created by academics	53
4.7	Instructional methods	55
	Conclusion	56
	Chapter 5	60
5.1	Introduction	60
5.1.1	Introductory programming	60
5.2	Method	61
	Exploratory focus group and online survey	61
5.2.1	Instrument development - exploratory phase.....	61
5.2.2	Focus Group	62
5.2.3	Data analysis – predictors of success	64
5.3	Exploratory phase results	65
5.4	Phase one Results	66
5.4.1	Children’s learning experiences.....	66
5.4.2	CS Concepts	67
5.4.3	Teaching process.....	68
5.4.4	Success Rates	69
5.4.5	Attitudes towards learning	70
5.4.6	Predictors of success	71
	Conclusion	72
5.5	Exploratory Case study in 4 primary schools	74
5.5.1	Phase two - Methods	74
5.5.2	The study.....	74
5.6	Measuring children’s progress – Quantifying the data	76
5.6.1	Data point one - Children’s progress through the course.....	76
5.6.2	Data point 2 – Teacher’s predictions of children’s progress	77
5.6.3	Data point 3 – Children’s motivation.....	78

5.6.4	Data point 4 – Children’s interests outside school as predictors of success	79
5.7	Results	81
5.7.1	Teacher’s role.....	81
5.7.2	Children’s learning experiences	81
5.7.3	CS curricula.....	83
5.7.4	Progress measures and scores	84
5.7.5	Progress scoring	85
5.7.6	Motivation levels (Data point 3)	86
5.7.7	Play interests Data point 4	88
5.8	Discussion/Future work	89
5.9	Limitations	91
Chapter 6	93
6.1	Quadrant C	93
6.2	Related Work	93
6.2.1	The Scottish Curriculum for Excellence	93
6.2.2	The Curriculum for Excellence computing science approach	94
6.2.3	CfE introductory programming in the primary classroom.....	95
6.2.4	The create-first approach.....	96
	Conclusion	98
Chapter 7	100
The research informed practice study	100
7.1	Introduction	100
7.2	Method	100
7.2.1	Aims and Objectives	100
7.2.2	Approach.....	101
7.2.3	Research Design.....	101
7.2.4	Setting	102
7.2.5	Participants	102
7.2.6	Materials.....	103

7.2.7	Procedure	105
7.2.8	Data	105
	Qualitative and quantitative data analysis.....	105
	Qualitative data collection and analysis.....	108
	Quantitative data collection and analysis.....	109
	Children’s enjoyment levels toys, games and activities.....	112
7.3	Results	113
7.3.1	Thematic analysis.....	113
7.3.2	The Ambassador’s reflections.....	115
7.3.3	KWL themes	117
7.3.4	Ambassador’s lesson plans	119
7.3.5	The Ambassadors’ journals.....	119
7.3.6	Classroom management	120
7.3.7	Children’s learning experiences.....	120
7.3.8	CS Concepts Results	127
7.3.9	Progress through the Curriculum	129
7.3.10	Measuring children’s Total scores across the three lessons:.....	131
7.3.11	Total scores across the three lessons by class:.....	133
7.3.12	Children’s enjoyment levels of play	134
7.4	Comparison of qualitative and quantitative results.....	136
7.4.1	Whole course level analysis	136
7.4.2	Lesson level.....	137
7.4.3	Question level	138
	Conclusion	138
7.4.4	Discussion/Future work	139
7.4.5	Limitations	142
	Chapter 8.....	145
8.1	Quadrant D:	145
8.2	Introduction	145

8.3	Related work	146
	Conclusion	151
Chapter 9	152
	Introductory programming studies in traditional primary school classrooms.	152
9.1	Introduction	152
9.2	Method	152
9.2.1	Aims and Objectives	152
9.2.2	Overarching Approach	152
9.2.3	Research Design for both studies	153
9.3	Study A – School B.....	154
9.3.1	School B – Setting.....	154
9.3.2	Participants	154
9.3.3	Materials.....	156
9.3.4	Procedure	156
9.3.5	Data analysis	158
9.3.6	Interrater reliability	158
9.4	Study A School B Results	159
9.4.1	Results	161
9.4.2	Planned learning	161
9.4.3	Results in progress	162
9.4.4	Qualitative data	164
9.4.5	Discussion/Future work	165
9.4.6	Threats to Validity/Limitations	166
9.5	Study B – School K.....	166
9.5.1	Aims and objectives	167
9.5.2	Approach.....	167
	Predictors of success (Independent Variable)	167
9.5.3	Study B School K Research Design.....	168
9.5.4	Participants.....	168

9.5.5	Materials.....	170
9.5.6	Procedure	170
9.5.7	Data analysis	171
	The role of the teacher.....	171
	The children’s learning process	171
	Course coverage	173
9.6	Results	182
9.6.1	The Role of the teacher Phase one and Phase two.	182
9.6.2	Children’s view of the level of difficulty (Comfort Level).....	188
9.6.3	Computing science concepts covered	189
9.6.4	Predictors of success	192
9.6.5	Children’s reported levels of enjoyment in toys, games and activities.....	199
9.7	Discussion.....	203
9.7.1	The ‘Role of the teacher’	203
9.7.2	The children’s learning process	204
9.7.3	The CS Concepts.....	205
9.7.4	Predictive variables	205
9.7.5	Added value of the intervention.....	205
9.7.6	Study limitations	207
Chapter 10	208
Discussion/Future work	208
10.1	Summary	208
10.1.1	Conclusion	210
10.2	RQ1:	211
10.3	RQ2:	211
10.4	RQ3:	212
Chapter 11	216
Epilogue	216
11.1	Background	216

11.2	Introduction	216
11.3	Computing Science Curriculum is expanding globally	217
11.4	Curriculum justifications.....	217
11.5	Primary School computing science (CS/CT) curriculum content.....	218
11.6	Computing science pedagogy at primary school level.....	220
11.7	Teacher’s PCK	220
11.8	Instructional methods	221
11.9	CS/CT pedagogical links with literacy	223
11.10	CS/CT links to other subjects.....	225
11.11	Conclusion	225
	References	227
	Appendices	244
	Appendix A Focus Group Pre-reading material.....	245
	Appendix B Focus Group questions	247
	Appendix C Quadrant B - Online survey	248
	Appendix D School A	252
	Appendix E - Quadrant B Assessment criteria	254
	Appendix F Curriculum for Excellence 3-step explained.....	256
	Appendix G Quadrant C Ambassador’s lesson plans	257
	Appendix H Ambassador’s full unedited lesson plans	259
	Appendix I Sample learning journal	275
	Appendix J Children’s Toys and Games survey	276
	Appendix J Sample table of results for P4 School B	277
	Appendix L School K Children’s view of level of difficulty during the planned learning.....	279
	Appendix M CfE Computing Science 3 Step overview.	280
	Glossary	281

Tables

Table 1 CS curriculum content and digital partnerships.....	17
Table 2 Universal (Soft) skills valued in sampled literature.....	26
Table 3 Phase one soft and hard skills codebook deductive codes	28
Table 4 Deductive codes from literature and inductive codes from initial analysis	29
Table 5 Themes arising from interviews with non-digital industries.	31
Table 6 CS studies in primary extracted from the systematic review.....	58
Table 7 Overview of CS studies in primary school contexts	59
Table 8 Participants' backgrounds	64
Table 9 Children's age taught and background of teacher.....	65
Table 10 Curriculum resources used for introductory programming.....	66
Table 11 CS concepts taught reported by 28% of respondents.....	67
Table 12 The range of teaching approaches.....	68
Table 13 Children's level of success.....	69
Table 14 Motivation levels during introductory programming courses.....	70
Table 15 Views of predictors of success	71
Table 16 Participants view of children's personality contributing to success in IP.	72
Table 17 Demographics of schools involved in exploratory case study	75
Table 18 Success criteria assessment banding for total scores	76
Table 19 Scoring for children expected and actual progress	77
Table 20 Motivation level scoring prior to starting the introductory course	78
Table 21 Motivation level scoring during the introductory course.....	78
Table 22 Motivation level scoring after the introductory course.....	79
Table 23 Categorisation of play interests for data point 4	80
Table 24 Results for each school	85
Table 25 Number of children achieving each band based on weekly SC scores.....	85
Table 26 Children's motivation levels.	86
Table 27 Gender and motivation level of band 5 children.....	87
Table 28 Regression output for highest achieving children and play interest.	88
Table 29 Categorisation of participants in the study.....	103
Table 30 KWL grid headings	105
Table 31 Sources of qualitative and quantitative data	107
Table 32 Shows the success criteria children use for self-assessment.....	110
Table 33 Sample self-assessment.....	111
Table 34 Survey questions organised by play categories.....	112

Table 35 Hierarchy of children’s reflections thematic codes. Learning dimension	115
Table 36 Hierarchy of thematic analysis codes CS concepts and Attitudes to learning ...	115
Table 37 The hierarchy of KWL codes.....	118
Table 38 Lesson learning intention and success criteria codes.....	128
Table 39 Mean and scores across cohort.....	131
Table 40 Total and question level scores(mean).....	132
Table 41 Question level scores(mean) by each lesson.....	133
Table 42 Mean performance for each class.....	133
Table 43 Mean performance for lesson one.....	133
Table 44 Mean performance lesson two	133
Table 45 Mean performance lesson three	133
Table 46 Children’s responses in the enjoyment survey showing by play category	134
Table 47 Correlation of toys games and age, gender total scores showing **	135
Table 48 Mean progress of each class.....	136
Table 49 Class progress lesson by lesson in descending order.....	137
Table 50 Children's SES status measured by SIMD deciles and data zones	155
Table 51 Learning outcomes	161
Table 52 Summary of statistics	163
Table 53: School B participants:	169
Table 54: Children grouped by reading and numeracy CfE levels.	170
Table 55 Overview of the traffic light self-assessment statement scoring and children’s view of the level of difficulty	172
Table 56 Scores applied to Code.org courses by CS concept.	173
Table 57 5 point scoring system	174
Table 58 Professional judgement of attainment in literacy and numeracy	175
Table 59 Questions from survey grouped by play categories	176
Table 60 Breakdown of each lesson used in the study and the concepts covered.	178
Table 61 Images from code.org showing content.	180
Table 62 Role of the teacher from observation notes	185
Table 63 Children's mental processes during phase one and phase two	187
Table 64 Children’s comfort levels in introductory programming	188
Table 65 Children’s progress through phase one and phase two	190
Table 66 Total no of chn. in Class A completing CS concepts in phase 1 and phase 2. ...	191
Table 67 Total number of children in Class completing CS concepts.....	192
Table 68 Class A Reading attainment groupings and progress in phase 1&2	194
Table 69 Class B Reading attainment and progress	195

Table 70 Class B Numeracy attainment of children and progress	196
Table 71 Children's comfort levels and reading attainment	198
Table 72 Reading and numeracy attainment and course progress	199
Table 73 Correlation matrix progress and levels of enjoyment in play	201
Table 74 Correlation children's progress and level of enjoyment in play categories	202
Table 75 Phase one and phase two tasks in relation to CfE 3 step approach.....	213

Figures

Figure 1: Overview of thesis quadrants	10
Figure 2 Process Thinking strategies in a primary classroom.	46
Figure 3 Shows the three-part study design for quadrant B.....	61
Figure 4 Shows contributing factors to learning in a primary classroom.	63
Figure 5 Screenshot showing Lego WeDo on the screen drag and drop interface	82
Figure 6 Image showing Lego WeDo robot.....	83
Figure 7 Research design focus to inform focus.....	101
Figure 8 Four study themes.....	108
Figure 9 The study evaluation framework	153
Figure 10 Boxplot comparing phase 1 and phase 2 distribution	163
Figure 11 Scatterplots of SIMD and phase 1 and phase 2 scores	164
Figure 12 Screenshot of predictive element in course C Loops.....	179
Figure 14 Code.org lessons matched to USA school system.....	189

Acknowledgements

I would like to express my gratitude to both of my supervisors Professor Quintin Cutts MBE and Dr Kristinn Hermansson for their invaluable support and insightful comments and suggestions through very different lenses at every stage of the thesis. Individually very different, collectively perfect.

A particular thanks to Quintin, for the most incredible journey into the enlightening world of academia, his impact on my personal and professional growth will be with me forever. For showing me how to find greatness in everyone through... listening.

To everyone connected to the University of Glasgow Centre for Computing Science Education. You are an amazing and talented group.

To my mum and dad, I wish you were here to see this! Thanks for instilling in me the value of education. To my fabulous children, Lauren, Eilidh, Isla and Ryan for just being them. Not forgetting Neil, Dominic and Beth. I love the seven of you very much.

To my big sister Cath, I treasure the emotional support, the space to write and the food. You and Ali really kept me going when it was most challenging. Thanks also to Marg and Sadie.

To Lindsey – I couldn't have achieved this without your wisdom and support. Thank you for navigating me through the challenges of academia and the demands of a PhD thesis.

To Sharon for that one coffee in Byres Rd that made this happen. George – thanks for all.

Finally, to Daisy, always at my side.

Declaration

I hereby declare that, except where specific references are made to the work of others, the contents of this document are original and have not been submitted, in whole or in part, for consideration for any other degree or qualification, in this or any other university. This doctoral thesis is the result of my own work, under the supervision of Prof. Quintin Cutts MBE and Dr. Kristinn Hermansson. Nothing included is the outcome of work done in collaboration, except where otherwise indicated within the text.

Publications

- Cole, E. and Cutts, Q., 2022, September. See, Talk, Explain and Prepare (STEP): The First STEP for Primary Teachers in Introductory Programming. In *Proceedings of the 2022 Conference on United Kingdom & Ireland Computing Education Research* (pp. 1-1).

Contribution: Lead author. This paper was written after the thesis and although it did not inform the thesis it represents the intervention used for scaffolding teachers link early literacy with computing science.

- Vivian, R., Quille, K., McGill, M. M., Falkner, K., Sentance, S., Barksdale, S., Busuttil, L., Cole, E., Liebe, C. and Maiorana, F. (2020) An International Pilot Study of K-12 Teachers' Computer Science Self-Esteem. In: 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20), Trondheim, Norway, 15-19 June 2020, pp. 117-123. ISBN 9781450368742 (doi: 10.1145/3341525.3387418)

Contribution: This paper is the third of 4 publications arising from participation in the ITISCE working group 2018. I Led the data gathering and analysis from Scotland's perspective.

- Falkner, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F., McGill, M. M. and Quille, K. (2019) An International Study Piloting the MEasuring TeacheR Enacted Computing Curriculum (METRECC) Instrument. In: 2019 ITiCSE Working Group Reports (ITiCSE-WGR '19), Aberdeen, Scotland, 15-17 July 2019, pp. 111-142. ISBN 9781450375672 (doi: 10.1145/3344429.3372505)

Contribution: This paper is the second of 4 publications arising from participation in the ITISCE working group 2018. I Led the data gathering and analysis from Scotland's perspective.

- Falkner, K., Quille, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F. and McGill, M. M. (2019) An International Comparison of K-12 Computer Science Education Intended and Enacted Curricula. In: 19th Koli Calling International

Conference on Computing Education Research (Koli Calling '19), Koli, Finland, 21-24 Nov 2019, ISBN 9781450377157 (doi: 10.1145/3364510.3364517)

Contribution: This paper is the second of 4 publications arising from participation in the ITISCE working group 2018. I led the data gathering and analysis from Scotland's perspective.

- Falkner, K., Quille, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F. and McGill, M. M. (2019) An International Benchmark Study of K-12 Computer Science Education in Schools. In: Innovation and Technology in Computer Science Education (ITiCSE '19), Aberdeen, Scotland, 15-17 July 2019, pp. 257-258. ISBN 9781450368957 (doi: 10.1145/3304221.3325535)

Contribution: This paper is the first of 4 publications arising from participation in the ITISCE working group 2018. Contributions for the paper involved representation from the four UK nations. I led the data gathering and analysis from Scotland's perspective. Although no direct relevance to the thesis findings, the work supported the development of data analysis.

- Cole, E. (2019) K-6 Introductory Programming: Why Early Years Learning through Play Matters. In: International Computing Education Research Conference (ICER '19), Toronto, ON, Canada, 12-14 Aug 2019, pp. 327-328. ISBN 9781450361859 (doi: 10.1145/3291279.3339442)

Contribution: I was the lead author of this paper as part of the ICER Doctoral Consortium. These findings contributed to Quadrant D chapter 9

- Cutts, Q., Patitsas, E., Cole, E., Donaldson, P., Alshaigy, B., Gutica, M., Hellas, A., Larraza-Mendiluze, E., McCartney, R. and Riedesel, C. (2018) Early Developmental Activities and Computing Proficiency. In: Proceedings of the 2017 ITiCSE Conference on Working Group Reports - ITiCSE-WGR '17, Bologna, Italy, 03-05 July 2017, pp. 140-157. ISBN 9781450356275 (doi: 10.1145/3174781.3174789)

Contribution: ITISCE working group joint lead Bologna 2017. Developed survey instrument, validated questions with early years experts. Indirect impact on statistical analysis in quadrant A, B, C and D.

- Cole, E. C. (2015) On Pre-requisite Skills for Universal Computational Thinking Education. In: ICER '15: International Computing Education Research Conference, Omaha, Nebraska, USA, 9-13 Aug 2015, pp. 253-254. ISBN 9781450336307 (doi: 10.1145/2787622.2787737)

Contribution: Lead and sole author of paper for conference Doctoral Consortium.

Ethics approval

Quadrant A: Ethics Approval: Ref - 300140047

Quadrant B: Ethics Approval: Ref - 400150096

Quadrant C and D: Ethics Approval - 400170048

Chapter 1

1.1 Introduction

The world is implementing CS

Digital technologies are everywhere with young children at an influential stage in their cognitive development forming attitudes about their use and value in society. It is likely, that children interact with digital toys, experience SMART technology controlling living environments and observe handheld devices used for a multitude of everyday functions including google answering many questions posed by their parents or carers and themselves. However, while our children are great “users’ of digital technologies, few understand how they work. There was a view that children need to know how digital technologies work. Computing science skills will enable improvements in existing products, develop new designs and innovations and develop products appropriate for underrepresented groups.

At this stage, politicians, policymakers and researchers publicised their support for computing science in formal education settings. However, in the primary classroom, it is not clear if primary teachers prioritised CSED within their already overcrowded curriculum. Importantly, in 2013 the computing science education community set the subject’s inclusion within the context of the United Nations Rights of the Child (UNRC) first principle and purpose of education; ‘Education is for a child’s personal development and participation in society’ (Schulte 2013). This is a justification that sits very well with those promoting its inclusion and underpins the political drive that focuses mainly on the economic benefits.

My personal motivation for this research project is as follows: In September 2006, I was appointed HM Inspector of Education (HMI) with a remit to evaluate learning and teaching across Scotland’s schools providing assurance for parents and Scottish ministers and to inform national policy. In 2014, my employers identified a gap in CS expertise within the organisation and I was assigned the remit of 3-18 computing science. It was at this point; I became aware of the CS policy roll out in formal education and limited insights on its value in primary education or best practice in implementation.

I have over 34 years’ experience in the primary education sector and evaluating learning and teaching across 3-18. My job on a weekly basis involves me sitting in classrooms, observing learning and teaching, interviewing teachers and children listening to their views

and taking account of quantitative data. This experience carries over well to the kind of studies that have been carried out in this thesis.

1.2 The thesis statement and research questions

The thesis statement is:

Recognising the importance and challenge of introductory programming for all primary pupils, there is value in balancing CS education research alongside primary teacher expertise in a comprehension-first pedagogy.

The thesis statement is explored using the following overarching research questions.

RQ1: Is there value in computing science education for all?

RQ2: What typical programming approaches are in place in traditional primary classrooms and how successful are they?

RQ3: Can a comprehension-first oriented introductory programming course be implemented successfully in a primary school classroom?

Abstract

Digital technologies are everywhere with young children at an influential stage in their cognitive development forming attitudes about their use and value in society. However, while our children are great “users” of digital technologies, few understand how they work. An argument evolved around 2014 that children need to know how digital technologies work and in doing so would develop valuable employability skills. It is now increasingly common for CS to be included in compulsory education for children as young as 3.

Throughout the thesis the terms computing science, computational thinking and introductory programming are used. All three terms are distinct in their own right and are defined as follows for the purpose of this thesis. Computing science (CS) is the overarching curricula experienced by children in a formal school setting. Introductory programming (IP) is component part of the computing science curricula that develops computational thinking (CT). Defining CT is outwith the scope of the research focus. However, CS, IP and CT are inextricably linked and are considered throughout the thesis.

CS in the thesis includes what is being taught the ‘content’ and how it is taught ‘pedagogy’. The thesis focuses on CS introductory programming and the use of visual

programming environments. Primary school teachers may use visual programming environments such as ‘Scratch’ to deliver the computing science curricula. Scratch lends itself to a ‘create-first’ approach whereby children explore the programming environment and producing scripts. Higher education institutes (HEI) promote program comprehension for introductory programming and students experienced examples of code before creating their own. A comprehension-first approach in a primary school context takes account of HEI program comprehension. The HEI comprehension pedagogy is modified to suit the developmental needs of young children.

This work follows an iterative approach, organising the theory into quadrants reflecting the evolving nature of education policy implementation. Each quadrant has a literature section and a study section.

1.3 Quadrant summaries

To support navigation through the thesis, each quadrant opens with a relevant work chapter followed by the study. Most literature cited is set within the relevant quadrant timeframe. However, more recent, and relevant literature to support statements and arguments is also included to maintain the relevance of the research. This following section provides a quadrant summary detailing the gap in literature and the study, participants, methods, and findings. For ease of reading, this summary is repeated at the start of each quadrant concluding with a visual representation in Fig 1 of the study as a whole.

1.3.1 Quadrant A: Is Computing Science Education of Value to All?

Quadrant A explores the territory and arguments in literature for CS in mandatory education with a focus on the primary school years age 5-12. The review identifies one critical gap, of the value of CS programming to all children in primary school whose post-school destination is outside digital industries? This quadrant explores the question of whether CS education is really needed for everyone and hence deserving of a place in the primary curriculum.

Motivation: Digital technologies are essential drivers to modern economies. CS curricula is argued by politicians and the CS community as an essential skill not only for those who want to be computer scientists but for everyone irrespective of their chosen career. As a

result, countries across the globe increasingly include K-12 computer science (CS) curricula frameworks drawing on collaborations dominated by digital industries and CS educators. The main goal of this quadrant is to explore the view that CS is of value for all. While the merits of a CS curricula may be obvious for future programmers and data scientists in quaternary¹ industries, (<https://www.britannica.com/technology/industry>) they may be very different for future decorators and bakers. Hence the question posed is, what relevance is CS to non-digital industries?

Objectives: Exploration of the importance of CS for everyone view led to the following research questions.

RQ 1: Is there value in computing science education for all?

Method: Key contacts from thirty-one non-digital industries were interviewed. Participants ranged from sole traders such as a sculptor to a large renewable energy company with over 20000 employees. As a category system, the deductive 'soft' and 'hard' skills codes are identified through literature. 'Soft' skills are generic employability skills and 'hard' skills are specific to the CS discipline. A multidisciplinary research team from psychology, education and computing employed a hybrid thematic analysis on the data. This flexible framework allows a focus while bridging across different understandings of terms.

1.3.2 Quadrant B: Typical CS approaches in primary education?

Motivation: At an unprecedented rate, many countries are introducing computer science (CS) into their K-12 curricula with variation in content and although CS is neither programming nor computer literacy, literature does suggest that most Primary school CS content features programming in the form of tasks related to computers (plugged) and off computer tasks (unplugged). Programming issues in tertiary settings are well-documented and its introduction to K-12 is complex, requires systemic change, teacher engagement, and development of significant resources.

¹ Quaternary industries as described by Encyclopaedia Britannica are concerned with information-based or knowledge-oriented products and services.

The computing science education research community rose to the challenge of supporting mandatory CS in schools. In 2011, an article published in the ACM inroads recognised the role of research in the CSED K-12 change process. The article set out a clear vision to support curriculum implementation (Barr & Stephenson, 2011) in relation to computational thinking (CT), which they state, has an important relationship with the CS practice of programming (National Research Council, 2010) (Denning 2003)

Collective efforts by the CS education research community led to the development of resources and the publication of studies based on theory-driven content and some data-driven experiments. These are included in more detail within the background reading section.

This study reviews literature on approaches in introductory programming tools and instructional methods. It looks briefly at the theory-driven literature and focuses on publications that test out introductory programming theory within the primary school context. Findings highlight a gap in studies set in typical primary school classrooms and that all data driven studies in classroom settings are supported by high adult student ratio led by the CS researcher. Studies involving children show variations in success rates. The related work section concludes that the studies, due to their set up are not replicable in typical classroom settings or easy to upscale.

Objectives: Given the significant challenge of training sufficient teachers (Brown & Sentence, 2014) in line with the implementation rates of CS in primary school curriculums there is a need to know how primary teachers without training implement the subject. This insight will support future work and resource development.

The study has two RQ:

RQ 2a: What typical introductory programming approaches are in place in traditional primary classrooms?

RQ 2b: How successful are typical programming approaches in traditional primary classrooms?

Methods: To answer the RQs a review of related work was undertaken and identified a gap in understanding typical approaches to CS education by primary teachers. The gap in literature led to a three part study with fieldwork involving CS educators and primary school children. Part one: A focus group of educators described their approaches to CS education in primary school. This information informed a survey issued more widely via

social media. From the survey, an exploratory study provided data from observations to support the survey and focus group findings. The exploratory study took place in 4 Scottish primary schools.

1.3.3 **Quadrant C: A comprehension-oriented approach led by CS experts**

Motivation: Quadrant B study explored primary teachers instruction design and outcomes for children in a traditional classroom setting. This prompted a focus on IP program comprehension in a primary context. At the time of writing (2017), in Scotland, Curriculum for Excellence (CfE) CS standards were issued. The revised CS curriculum in Scotland moves from a content creation-oriented curriculum to a concept driven comprehension-oriented approach. In essence, the CfE CS curriculum promotes comprehension of code before writing code. This added dimension meant it was therefore unlikely to find teachers in primary school with full CSED or IP knowledge or CfE CS knowledge experts with a deep understanding of primary school education. Increasingly insights from literature led by the CSED research community continue to be published. Therefore, it was deemed appropriate, to explore and evaluate the comprehension-first oriented approach in the primary classroom.

Objectives: The study builds on recommendations in quadrant B and explores the implementation of the Curriculum for Excellence (CfE) Computing Science (CS) in formal primary education. It also begins to consider predictors of success in introductory programming.

RQ 3: Can a comprehension-first oriented programming course be implemented successfully in a primary classroom?

Method: A study was planned using adapted materials created by the Ambassadors who are CS students learning about CS education. 2 CS students planned and implemented materials to 3 cross age classrooms in 2 schools. Using the CfE 3- step approach, 2 Ambassadors with no primary school experience, plan and implement a 3 x 1 hour IP course in 2 schools. The lesson design aligns with a comprehension-first oriented approach. Step 1: Children understand programming concepts, step 2: children understand the tools for programming such as Scratch a drag and drop editor or Kodu game creation tools. Step 3 children use the tools to build computational solutions. The planned learning aims to develop children's knowledge of the introductory programming course underlying

concepts. The lessons planned to show children the relationship between real life processes and those represented using tools and languages that use CS concepts. Children are motivated by the range of activity based learning and instructional design.

A convergent parallel mixed methods approach analyses the Ambassadors and children's reflections alongside quantitative data of known predictors of success. Age; gender and interests in activities outside formal learning. Each lesson was observed by the classroom teachers.

1.3.4 Quadrant D: A comprehension-first approach led by primary teachers?

Motivation

CSED literature continues to increase focusing on challenges introducing basic programming concepts to young children. However, not surprisingly, this increased interest brings a lack of agreement as to the best approach. A common starting point in many countries and schools is to teach the Scratch computer programming language to children. Block-based programming environments are motivational, they lend themselves to create-first, an approach that brings quick rewards but some authors consider their use may lead to children not understanding the underlying concepts.

This quadrant builds on the findings from quadrant C and aim to involve primary teachers as active participants in the development of meaningful approaches that align with primary school methods more generally. By 2018, the Scottish curriculum underpinned by the comprehension-first approach had been in place for almost a year. This comprehension-first oriented approach in primary contexts is also emerging in literature. Findings in Quadrant C show that a comprehension-oriented approach to introductory programming using visual programming languages aligns with the Scottish CfE CS curriculum. The approach impacts on children's progress and their levels of motivation. However, the study was led by the CS expert and observed by the classroom teacher. Gaps in literature show studies in primary school contexts continue to be researcher led with limited input from practising primary school teachers.

Objectives:

RQ 3: Can a comprehension-first oriented programming course be implemented successfully in a primary school.

RQ 3a: What instructional designs do primary teachers adopt in introductory programming?

Method: This study uses a researcher-practitioner collaborative approach with a focus on children's outcomes in an introductory programming course implemented in a traditional primary classroom. Two class teachers from two schools (4 teachers in total) each plan and deliver a 6-x 1 hour introductory programming course to their regular class. Lessons 1-3 they apply their own approaches and lessons 4-6 they apply a comprehension-first to their typical methods. In lessons 1-3, teachers choose their own resources and approaches. In lesson 4-6 the primary teachers approach was modified using a comprehension-first approach.

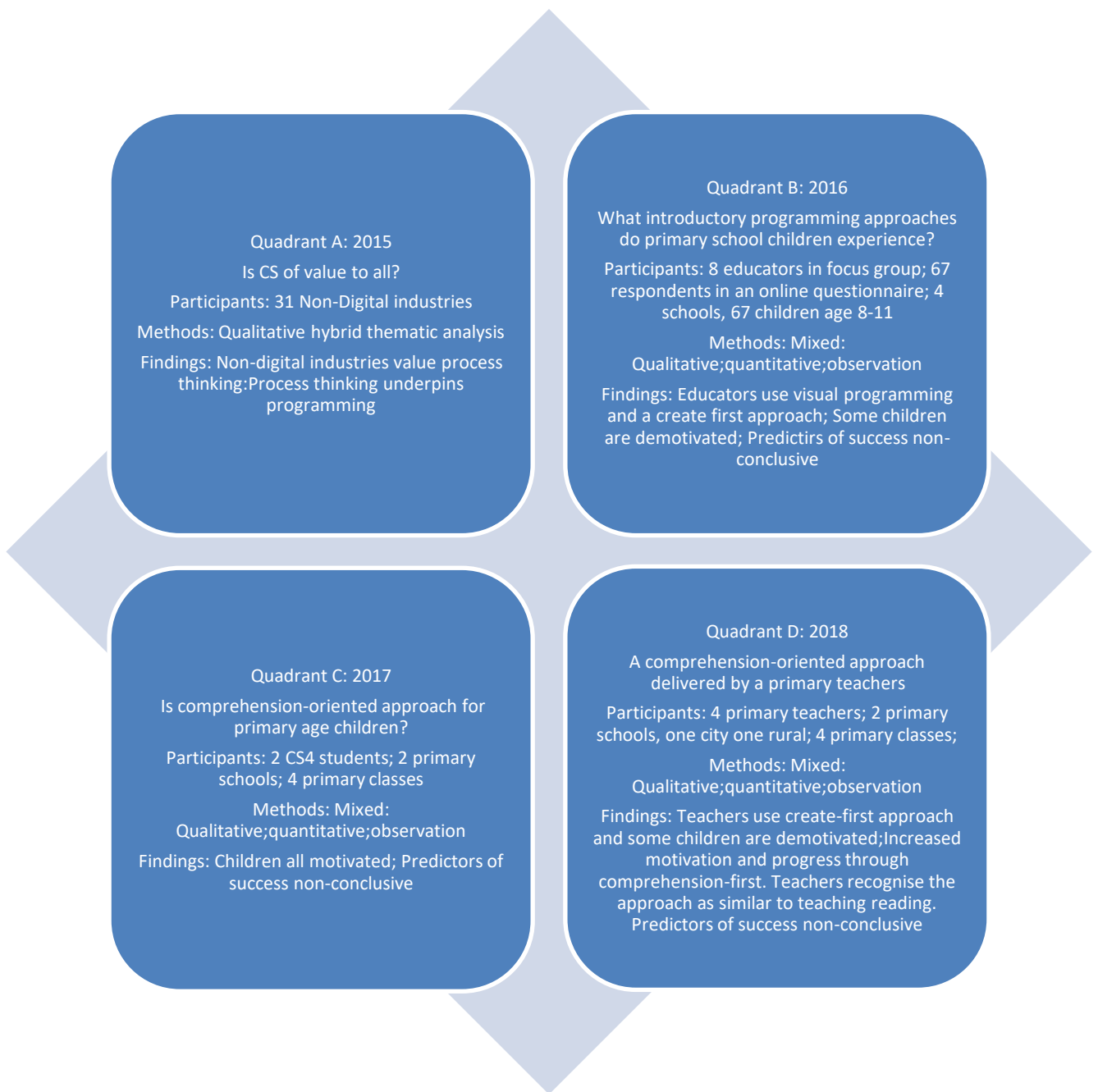


Figure 1: Overview of thesis quadrants

Quadrant A: Computing Science for All?

The thesis quadrants reflect the evolutionary nature of CS education implementation in formal primary school education. The research starts in 2015 with quadrant A comprising of two chapters (chapter two and chapter three) and lays out the territory for a computing science curricula in formal primary education.

Chapter 2

2.1 Quadrant A: Chapter 2

Introduction

At an unprecedented rate, many countries are introducing computing into their K-12 curricula. Not surprisingly, given the relative newness of computing science as a school subject, there are differences in countries implementation rates and content (Falkner, K. *et al.* 2019). However, politicians and the CS community drive on the importance of CS education for all, not just those specialising in CS careers arguably underpins much of the expansion. The view that CS education develops important skills for the digital world and the aspiration that children are creators and not just consumers of digital technologies is now widely accepted. While the authors agree that computing science is important in compulsory education, effective integration brings issues and challenges such as what should be taught and how it should be taught.

The CS K-12 debate is live with numerous studies and theories on its applications within compulsory education. While some CS curricula has been recently introduced into a number of countries, it is acknowledged that CS education is also entrenched in many eastern European countries. However, around the time of 2006 and the increased acceptance of CS Ed for all, the interest to introduce CSED in school curricula widened. As a result of the interest, funding initiatives backed by the tech giants emerged and curricular frameworks developed in collaboration with the CSED community and the technology corporations such as Microsoft and Google.

This study focuses on the concept of CS developing a set of skills beyond solely the digital domain. It explores this view given K-12 curricula typically depends on CS educators and CS-focused companies. If CSED K-12 develops skills valuable for all, then for authenticity there is a need to explore this claim from outwith the digital domain. Despite the gradual increase since 2013 on CSED research reviews (Tang et al., 2020), none, as far as the authors believe, show the voice of non-digital industries in their development. In addition, known views of non-digital industries valuable employability skills have not been reviewed through the lens of CS. It is worth noting that this study does not disagree with the digital dominance; however, it is worth gaining further insight in what is of value in other contexts.

While setting the research questions, the broad belief that compulsory education is about more than creating workers and employees is stressed. However, with the increasing

demand for employability, education in schools (Taguma et al., 2018) computing science can play in an important role.

RQ1: Is there value in computing science education for all?

To answer the research questions, 31 key contacts from non-digital industries ranging from sole traders such as a sculptor to large companies with over 250 employees specialising in renewable energy are interviewed. The interview focus is the qualities of the most effective employees. Patterns in the data are analysed to identify themes relating to the RQ. For reliability, a multidisciplinary research team from psychology, education and computing and employ a hybrid thematic analysis on the data. Use of a flexible framework for analysis maintains the focus on CS while allowing the data to derive themes with little pre-determination. As a category system for the deductive codes 'soft' and 'hard' skills are identified through literature. 'soft' skills are generic employability skills of value to all jobs and 'hard' skills are specific to the CS discipline.

2.2 Background and related work

2.2.1 Computing Science Education - A skill for All?

The role of education in the digital skills dilemma is considered as an option to grow more talent through formal education pathways for younger children. However, “supplying the digital skills pipeline” is not a statement that sits well with most primary teachers view of education who typically value developing a child as a whole person through skills for learning, life and work. On the other hand, when the phrase “Computational thinking” (CT) entered the debate, an interest outwith the CS community was sparked. CT aligns very well with the more holistic aims of primary education.

In the 80s Seymour Papert's pioneering ideas in the *Mindstorms* book (Wooster and Papert 1982) promoted the need to think computationally. At a time when children possibly had little interaction with a computer, he insightfully raised issues that 'the computer is being used to program the child' and the need for children to learn to program. Papert's view received a fresh 21st century perspective through Jeanette Wing's seminal paper (Wing 2008). The concept of thinking like a computer scientist, or CT resonated with educators, education researchers and policy makers. Her “rallying cry” argued for adding this competency to every child's analytical ability claiming that this type of problem-solving represents, in her view, a universally applicable skill set. Not only can computer scientists understand a problem and provide a solution, but they can create the solution for computer

technology to solve. Therefore, an age appropriate CSED providing a problem solving approach added to a child's existing problem-solving toolbox, would be of great value. Fast forward to 2015 and justifications for CS K-12 inclusion include Economic and workforce development; equity and social justice; competencies and literacies; citizenship and civic life; scientific, technological and social innovation; school improvement and reform; and fun, fulfilment and personal agency. Most definitely if it can achieve these then it surely is of value to 'ALL'. There has been some vigorous and unresolved debate about CT (Bocconi, et al., 2016) however, this review of background work focuses more on the justification for CS Education as of benefit for all. It is noted that the claim occurred at a low point in enrolment in CS programmes. Therefore, a claim that computing education not just for programmers and hardware specialists, but for everyone, is possibly an important tool used by CS educators in their attempts to address the enrolment issue. Paradoxically, the enrolment crisis was developing just as CS was becoming an increasingly strong economic driver and governments woke up to the huge disconnect between their education pipelines and the needs of industry, spawning the generation of reports on the issue. Concurrently, the resurgence in CSED resulted in a few countries reviewing their K-12 skills for work programmes as a driver for curricula reform (Kashefpakdel et al. 2019).

2.2.2 CS for "ALL" – The Journey

For decades, prior to the CS for all movement, Israel and many eastern European countries were offering rigorous CS courses. However, the expansion of CSED K-12 in such large scale while welcomed was not anticipated (Tang, Chou and Tsai 2020). The goal of accessible CSED with children moving from using software (ICT) to creators with the skills to create new technologies gained increased interest and funding initiatives backed by the tech giants. In 2007, Microsoft granted Carnegie Mellon University 1.5 million dollars to establish a research and study centre. In 2011- 2013, NZ introduced new standards. The desire for CSED inclusion gathered momentum with Microsoft, Facebook, Amazon, the Infosys Foundation and Google sponsor Code.org developing a vision that every student in K-12 has the opportunity to learn computer science. In 2016 President Obama announced 4 billion Computer Science for All initiative. In England, £84 million to upskills CS teachers and a national centre for computing education that also supports resources for primary and secondary school age children.

The appetite to implement CS quickly means few academic studies experimenting with new ideas were upscaled sufficiently well for national implementation strategies.

Numerous computing science curricula now incorporate computational thinking in some

form. However, there remains no clear explanation of the complicated relationship of CT and CSED with an often disproportionate focus on programming (Pollak and Ebner 2019). How then does this confusion impact on our primary teachers and ultimately the youngest children in the education system?

2.2.3 CS curricula collaborations with digital industries.

The background review so far lays out the argument for CS in mandatory education, the increased funding initiatives, and the expansion in computer science education. The benefits of this expansion for the CS community and digital industries are clear. This study is however, interested to determine whether computing science education could be of more value to these directly computing contexts, as had been originally envisaged. The strong industry partnerships evident in, for example, the K-12 CS Framework, Learning to Code in China and BCS Barefoot UK sit with highly influential digital industries such as Microsoft, Amazon, Facebook and Google. The Australian Computing Academy (ACA) based in the University of Sydney's School of IT in the faculty of Engineering and Information Technologies received 10 million Australian dollars to provide Australian teachers with educational resources and professional development necessary to deliver the Australian Digital Technologies curriculum. In addition, Google Education mapped resources to the Australian curriculum². In New Zealand, the Curriculum advisory panel created curriculum content in light of feedback from the Digital Technologies & Hangarau Matihiko curriculum consultation (Curriculum Advisory Group 2017) which consists of digital industry partners. In curricular resources readily available online, it is apparent that, where strong industry partnerships are evident, these sit with highly influential digital industries, an oversight not yet addressed in the CSED literature.

To summarise the potential issue is, if the concept of CSED and CT in particular is a set of skills valuable beyond the digital domain, the lack of clear support for "CS for All" curricula from academics and non-digital industries seems surprising and at odds with the rigour of claims expected within the broader academic arena. Despite the gradual increase since 2013 on CSED research reviews (Tang, Chou, et al., 2020) none, show the voice of

² Digital Technologies Hub. Available at <https://www.digitaltechnologieshub.edu.au/> Accessed 24 December 2021

non-digital industries in this development. There are many advantages for digital collaborations and moving outside the digital domain will build on this knowledge. However, it does call for a fresh look at career education in schools through the lens of CSED skills.

Initiative	Country	CS-programming content specified in curricula or resource	Dig* ind.*
K-12 CS (Stephenson et al 2012)	USA	Computational problems, abstractions, computational artefacts	*
Learning to Code (Asia News 2016)	China	Algorithms – CS principles that underpin all digital technologies, core programming concepts	*
Barefoot (British Telecom 2016)	UK	Abstractions, logic, algorithm, data representation	*
Curriculum (ACARA 2014)	AUS	Problem solving, designing and use of algorithms.	
Curriculum (Bargury et al 2012)	ISR	Algorithmic thinking	*
Curriculum (Vipul 2019)	IND	Algorithm, decomposition, pattern recognition, abstractions.	
Nordic Countries (Bocconi 2018)		Abstractions, algorithmic thinking, automation, decomposition, generalisation	
Dig. Tech. & Hangarau Matihiko curr (Curriculum Advisory Group 2017)	NZ	Decomposition, algorithmic thinking, debugging.	

Table 1 CS curriculum content and digital partnerships

Chapter 3

The Study

Introduction

Before undertaking the study, the broad belief that compulsory education is more than creating workers and employees must be emphasised. However, with the increasing demand for employability enrichment activities in schools, the important role that CS can play in equipping children and young people for the working aspect of their lives is recognised, and hence the importance of this question.

3.1 Methods

3.1.1 Aims and objectives

RQ1: Is there value in computing science education for all?

This study aims to bring insights to the view that CS is of value to everyone, irrespective of their chosen career and therefore is of value to all in formal education. Thirty-five non-digital industry representatives are interviewed in a two-stage study. Interview recordings were transcribed, and a hybrid thematic analysis was carried out to determine recurring themes relevant to CT across the interviews.

3.1.2 Approach

The qualitative approach of the study was informed by Shutz theory of social phenomenology (Natanson, 1968) as both a philosophical framework and methodology. Schutz's theory emphasizes the spatial and temporal aspects of experience and social relationships. Social phenomenology takes the view that people living in the world of daily life can ascribe meaning to a situation and then make judgments. Therefore, the topic for interpretation is the subjective meaning of the non-digital industry participants experience. The descriptive and interpretive approach of the study is modelled on the hybrid thematic analysis described by Fereday and Muir-Cochrane (2006). The combined technique of inductive and deductive hybrid thematic analysis is applied because of the newness of the topic.

3.1.3 Research Design

The hybrid thematic analysis is a descriptive research design. From literature available at the time, research in the value of CS for primary school age children is limited. The scope of employability skills studies is often limited to generic transferable skills. A hybrid

approach is advantageous for this type of data collection because it integrates pre-determined themes with those that emerge from the data. The research design uses qualitative methods, most suited for this type of exploratory study to understand an issue which currently does not have many theoretical explanations. Samples in qualitative research tend to be small in order to support the depth of case-oriented analysis that is fundamental to this mode of inquiry. Additionally, qualitative samples are purposive, that is, selected by virtue of their capacity to provide richly-textured information, relevant to the phenomenon under investigation. Select information rich cases. Semi-structured interviews (Charmaz 2006) (Holloway and Brown 2016) generate detailed insights answering the RQ. Lincoln and Guba's (1985) five stages for trustworthiness (3.2.7) namely, credibility, transferability, dependability, and conformability (3.2.7) are followed. More information about the five stages can be found in section 3.2.7.

Writing a codebook in thematic analysis enables researchers to search for topics across data and identify patterns for a given study. The goal was to create conceptually concise deductive codes. Creating the codebook is explained in more detail later on in this chapter. The deductive codebook includes 'soft' and 'hard' skill themes. 'Soft' skill codes from general employability skills and 'hard' skills of computer science from K-12 CS curricula.

3.1.4 Setting

The pilot phase takes place in the University of Glasgow school of computing science. After evaluating the pilot phase, the second phase takes place at the participants place of work.

3.1.5 Participants

The study in total involves key personnel from thirty-five non-digital industries. Ethics approval was obtained before recruiting participants. The 35 participants employed in non-digital industries (Table 2) include 22 males and 13 females between 30 and 65 years. The Participants ranged from sole traders such as a sculptor to a large renewable energy company with over 20000 employees. (Table 3). With the exception of two sole traders, all thirty-five participants had a line management role across a broad range of industries. Participants experience and engagement with employees provide a representative sample with deep and valuable insight for the study.

Stage one, pilot phase participants were selected through a nationally recognised voluntary organisation for employed graduate volunteers who support education in schools for the

four pilot phase key informants. The voluntary organisation communicated the study via email and participants self-selected to participate in a one hour pilot phase interview. Participants for stage two involve a further twenty-one participants from the same national organisation as the pilot stage. Using snowballing techniques an additional ten participants were recruited. All participants had a line management role across a broad range of industries. Their experience and engagement with employees provide a representative sample with deep and valuable insight for our study.

Industry Type	Number interviewed	Industry Type	Number interviewed	
Accountancy	2	Health protection	1	
Architect	1	Employment agency	1	
Aviation	2	Painter and decorator	2	
Café owner	1	Photography	1	
Cleaning	1	Renewable energy	3	
Commerce	1	Retail	2	
Confectionery	1	Sculptor	1	
Construction	3	Soft drink bottling plant	1	
Engineering	5	Solicitor	1	
Food and drink	4	Sport	1	
Total	19		12	35

Table 2. No. participants by industry (4 removed post pilot phase)

Participants range from sole traders such as a sculptor to a large company with over 20000 employees.

Classification	Number of employees	Number of participants
Micro	<10	10
Small	<50	8
Medium	<250	8
Large	>250	9

Table 3. No. of participants by company size

3.1.6 Materials

Good questions in qualitative research should be open-ended and clear to the participants¹. In order to achieve clarity for the focus group participants and stimulate discussion, pre-reading material in advance of the phase one interviews.

CS curricula content varies and therefore defining broad CS characteristics requires careful consideration. However, an emphasis in CS K-12 curricula content is apparent and evident in the content and resources sampled online within the Australia, China Asia, implementing New Zealand, the Nordic countries, United Kingdom Barefoot and the United States of America new frameworks. The pre-reading material content was developed by two CS academic with over 30 year's experience each in delivering HEI CS courses. They based their view on their experience, Stone (1970) and Denning (1989) and existing CS curricula materials. It is worth noting at this stage that the CS curricula frameworks available online draw heavily on programming vocabulary.

The aim of the pre-reading material is to capture transferable CS skills using accessible information for participants. The CS academics attempt to use terminology that the participants could connect with, and recognise that vocabulary may be interpreted differently. For example, algorithm is very significantly a CS word and words like abstraction, decomposition, generalisation, pattern matching can be defined differently across disciplines. while maintaining the integrity of the subject (Appendix A).

In summary, the pre-reading material proposes that computing is an information process. It is a process involving the acquisition and manipulation of information through the information processor. They suggest that, before the advent of mechanical computers, a

computer could be a human who followed well-defined process descriptions in order to manipulate information for some purpose. This human thinking process is also required in the computing domain in order to work out how to represent problems or tasks of interest using the information and processing mechanisms of a computer.

In addition, it is worth noting that human's first attempt at modelling the problem/task using the computer processing mechanisms may not be successful. In the case where the modelling is unsuccessful, there is a need for the human to analyse carefully the model alongside the problem and make improvements.

Devising the correct computational process for the real world is a highly complex human activity. The CS experts propose that process thinking is required for developing computational processes using the information processing mechanisms of a computer to represent real-world processes. The overarching conclusion of the 'experts' viewpoint is that CS is the study of computation, automation and information and the systematic study of algorithmic processes that describe and transform information through modelling and reasoning. The pre-reading material is summarised under four summary headings of *information, process, modelling* and *reasoning*.

- ***information***: The ability to identify and make sense of information. Recognising attributes required in order to specify a particular classification.
- ***process***: The ability to create data representation of process using algorithmic thinking
- ***modelling*** of processes and information. The ability of modelling information through abstractions, and
- ***reasoning*** (about the models of processes and information): The ability to test, debug and refine solutions

3.1.7 Procedure

The pilot phase

The study is organised into a pilot stage with a focus group and pre-reading material for four participants followed by the main study with thirty-one participants from non-digital industries.

The pilot phase was led by a computing science expert within the university of Glasgow and held in the school of computing science. This phase tested the appropriateness of questions and viability of research. The pre-reading material issued in advance of the interviews aimed to inform participants about the structure of the session, the questions and focus (Morgan, D.I, 2018). However, the content was too detailed and some participants

paid more attention to the material than others. It was clear that the pre-reading material and setting negatively influenced the discussion. Despite several attempts to focus the conversation on general employability skills, the conversation focused primarily on computers and the use of software packages such as Microsoft office word processing. In addition, one voice dominated the discussions.

The evaluation of the pilot phase led to the following important changes for the main study. The study moved from a focus group model (MacIntosh 1999) to individual interviews and participants would be interviewed in their place of work. Neutral locations can be helpful for avoiding either negative or positive associations with a particular site or building (Powell, Single, et al, 1996). It was felt that interviews taking place in the university were influencing responses. Each interview in phase two would be led by the thesis author skilled in in-depth interviewing techniques.

The participants information sheet was reduced to the minimum required for ethical approval and all references to computing science curriculum removed. The discussion prompts pre-reading material was replaced with two open questions 'describe your industry' and 'what are the qualities of your most effective employees'.

After the pilot phase were addressed and to test out the changes, one of the original participants was interviewed individually in his own workplace. He was positive about the improvements; felt he could speak in depth about his experience and that on reflection he was more relaxed participating in his place of work than the focus group at the university. Although individual interviews are time consuming, the deep description gained through the one-to-one approach justified this method of data capturing (Dziallas and Fincher, 2019).

Phase two interviews took place in 2016 over a period of 6 weeks, they were audio recorded lasting between forty-five minutes to one hour. Audio recordings for the main study were transcribed before analysed.

3.1.8 Data Collection and Analysis

The sampling unit is the oral responses on employees most effective qualities. To evaluate this complex topic, a hybrid thematic analysis is used.

Data collection methods

Two-stage interpretative consisting of a pilot stage to test the appropriateness of questions and viability of research study. Interviews are recorded and transcribed. The pilot phase, held in the school of computing science, included two observers, one independent from the research team, to highlight issues. The review of the pilot phase led to changes in the pre-reading material and location of the interviews.

Trustworthiness

Trustworthiness of results is key to high quality qualitative research, and researchers applied a five-stage framework of credibility, confirmability, dependability, transferability, and authenticity (Lincoln and Guba, 1985). A form of independent analysis and moderation of the research was implemented through individuals from the school of education and the school of computing science. The moderators had clear quality assurance roles increasing the reliability of methods. Each moderator's expertise across disciplines impacted positively on the process. **Confirmability** was achieved through the moderators being independent from the interviewing process. Transcripts were transcribed independently and proportionately cross-marked. The coding book themes were updated informed by the interview content, literature as detailed previously, and participants own experience.

For **dependability**, one of the moderators was not familiar with the CS discipline and independent of the data capturing process. She quality-assured the interviewing techniques employed in all transcripts. She familiarised herself with the transcripts and her first task involved reviewing each transcript for interviewer bias. Any interviews with bias would be discounted from the study. The moderator was confident that the author's questioning and probing techniques did not lead the interviewees and that any paraphrasing used by the interviewer to confirm participants' views was free from bias. She noted that, in the small number of occasions where the interviewer's probing questions, or summaries to clarify participants' responses, appeared to be leading, she concluded that the confidence in the interviewees' responses discounted any concern on this point.

A hybrid approach of inductive and deductive thematic analysis (Fereday and Muir, 2006) was employed and applied to field notes and transcripts of the thirty-one non-digital industry interviews.

The step-by-step (Nowell et al. 2017) approach for conducting a trustworthy thematic analysis guided our approach. From a deductive thematic analysis approach, the code book was initially populated with the soft skills determined to be relevant to all jobs and expected to be found in every interview. It was uncertain which industry-specific hard-skills would emerge as important; hence an inductive thematic analysis approach was

employed. For **credibility**, investigator triangulation was applied whereby more than one researcher was involved in the analyses. This also included researchers across three different disciplines. The moderators randomly selected and analysed ten sample transcripts in order to further populate the code book. The sample transcripts were reviewed to check emerging themes. Following this sampling, there was a discussion about the themes inductively arising from the data, and the initial coding template extended. A further two sample transcripts were coded and themes noted that they found difficult to employ, aspects of the texts not covered by the code book and any other relevant issues. Discussions of such observations then lead to further revisions of the themes. The code book was then revised and everyone met again, on one more occasion, to independently code another transcript. This allowed a final thematic template, upon which all agreed, to be created and this template was used to thematically code all of the transcripts. For moderation purposes, individuals cross analysed one third of the transcripts and using NVIVO³ applied Kappa statistical measure of inter-rater reliability (Cohen, 1960) which returned a value of 0.82. This high value can be interpreted as strong agreement. Finally, for **transferability** the study design aims to clarify the social and cultural contexts surrounding the data collection.

3.1.9 Soft skills

Literature suggests organisational performance in a fiercely competitive economy depends on a skilled workforce. In addition, over time globalisation and advancement in technology impacted on essential employability skills (Patacsil and Tablatin 2017). Defining general employability skills for the study appears challenging due to discipline variations and a lack of consensus on what constitutes employability skills frameworks (Bansal, Aggarwal and Singh 2020). (Potgieter and Coetzee 2013) (Griffiths et al. 2018). In literature, the range of terms for similar skills such as, core skills, key skills, essential skills and basic skills is wide ranging. However, one recurring categorisation within employability frameworks and appropriate for the scope of this study is content organised into universally applicable ‘soft’ employability skills and discipline specific ‘hard skills’ (Schaberg 2019).

³ <https://lumivero.com/products/nvivo>

The variations in valuable employability skills definitions is acknowledged. However, for the purposes of the thesis it is important that an attempt to classify the skills is made without them becoming a barrier and narrowing the focus of the work. The deductive codebook ‘soft skills’ for the thematic analysis was created through a desk exercise interrogating K-12 general employability (soft) skills of the World Economic Forum (World Economic Forum 2016), Employability skills K-12 (Sermsuk, Triwichtkhun and Wongwanich, 2014), Employability skills (Curtis and McKenzie 2001) and D2N2 (Hutchinson, et al, 2015). For the purposes of this study, the ‘soft skills’ are organised into the following themes of emotional intelligence, motivation, and flexibility.

	Country	Summary of soft skills valued
World Economic Forum (World Economic Forum, 2016)		Motivated, agile workforce
Employability skills K-12 (Sermsuk, Triwichtkhun and Wongwanich, 2014)	UK	Communication, elf-management, teamwork, creativity, informed, confidence, drive, resilience, reflection.
Employability skills (Curtis and McKenzie 2001)	AUS	Communication, teamwork, creativity, time management
D2N2 (Hutchinson, et al, 2015)	UK	Self-motivated, self-assured, aspirational, informed, teamwork

Table 2 Universal (Soft) skills valued in sampled literature

3.1.10 Hard skills

Extracting school content from academic disciplines to determine a subject’s technical or hard skill is challenging and typically structured around the major concepts and principles of the discipline. In addition, the newness of CS adoption in primary schools is another level of complexity. This section does not aim to add to the confusion, instead it revisits the fundamental basics of understanding how computers work alongside computer scientists problem solving.

Computer scientists have a range of skills associated with hardware, networks, cyber, HCI/design. In addition, they write software to make computers do new things or accomplish tasks more efficiently. In order to achieve this, they require a unique approach

to thinking that involves computation. Essentially, a computer codes i.e., changes information, stores information, uses information, and produces an output, retrieves info. Interestingly, Information Process Theory (IPT) (Dambre *et al.* 2012) is the computer metaphor used by psychologists to describe how the human mind works. Information process, is arguably carried out by humans as well as computers. This is hardly surprising when the original computers were humans carrying out complex mathematical tasks or procedures. The blend of computing and human activities suggests that the twin concepts of information and process appear central to computing and to the human mind in a range of contexts.

There is an acceptance that K-12 CS curricula content varies and therefore defining broad CS characteristics requires careful consideration. However, the focus group pre-reading material was based on expert knowledge from CS academics and available curricula frameworks. The available CS frameworks draw heavily on programming vocabulary and align with the 'hard' discipline specific of *information process, modelling* and *reasoning* deductive codes expanded below.

Information process is the basic foundation of CS and a core element of technical subject content or 'hard' skill. It links to the CS specific vocabulary used in curricular content. Take for example algorithms, the algorithm process should have a well-defined starting point, operating on information and process through discrete steps, reaching a clear completion point in a finite time, accomplishing a well-defined task; the description should be unambiguous, and have a level of generality in it such that it can be used to solve a whole class of tasks. There are many shared characteristics too, such as parallel and communicating processes perhaps with shared resources; they may be hidden or visible activities; there needs to be some form of executing agent that drives the process forward, whether that is human or mechanical. It also underpins the typical terms of decomposition, generalisation and pattern making which arguably sit in other disciplines although used in CS *modelling* and *reasoning*.

Finally, the essence of CS programming 'hard' skills is taken from the material developed by the CS experts (see section 3.1.1) and Appendix A. They are classified as *information process; modelling and reason*.

Phase One	Deductive codes	Phase One	Deductive codes
Soft Skills	1.0	Hard skills	2.0
Emotional intelligence	1.1	Information	2.1
Motivation	1.2	Process	2.2
Flexibility	1.3	Modelling	2.3
		Reasoning	2.4

Employees attributes: Soft and hard skills

Table 3 Phase one soft and hard skills codebook deductive codes

Data processing and diagnostics

Evaluation of the pilot phase resulted in changes in the main study design due to the influence of the pre-reading materials and place of interview. Therefore, pilot stage data removed.

The 'soft skills' and 'hard skills' framing, therefore, is appropriate within this context enabling accessible considerations of the value of CS outside the discipline. For the first analysis, the code book separates information process into two distinct codes. Once familiar with the summary heading 'soft skills' and 'hard skills codes (Table 4), the code book was applied to the transcripts. Coding of the transcripts was undertaken line by line with interests and questions highlighted. The inducted codes from the open coding informed the revised version of the codebook.

Phase 2 codebook

Soft skills	Hard skills	Support structures
Emotional intelligence <i>Deductive</i>	Information <i>Deductive</i>	Sharing Good Practice <i>Inductive</i>
Motivation <i>Deductive</i>	Process <i>Deductive</i>	Evaluations <i>Inductive</i>
Flexibility <i>Deductive</i>	Process following <i>Inductive</i>	Continuous improvement <i>Inductive</i>
Self-awareness <i>Inductive</i>	Process creating <i>Inductive</i>	Management structures <i>Inductive</i>
Organisation <i>Inductive</i>	Modelling <i>Deductive</i>	Systems <i>Inductive</i>
Learning skills <i>Inductive</i>	Reasoning <i>Deductive</i>	Technology <i>Inductive</i>

Table 4 Deductive codes from literature and inductive codes from initial analysis

Limitations

It is important to note that the purpose of the pre-reading material for participants is to stimulate focused discussions with the focus group participants. The CS 'experts' view aligns with Denning P.J (2005) and Stone (1972), It is acknowledged that there are other models and sets of principles that could be considered. In addition, the terms used may

have wider implications for those with a CS background. However, defining CS is out of scope for the thesis and could be included for future work.

3.2 Results

In reviewing the coding of all transcripts, patterns were identified, and themes generated capturing the underlying ideas. At this stage in the analysis, some codes were discarded due to their vagueness or because they did not appear very often in the data; there were also instances where codes were combined into final themes for the results.

Using the methods described previously in the paper, twelve new codes of value emerged from the data. Three of the six final codes relate to the 'soft' skills found previously the general employability literature.

In reviewing the coding of all transcripts and considering the weight of evidence found for each code, five main themes emerged from the data. One theme relating to the soft skills and four process related. The high value that literature (Kashefpakdel et al, 2018) places on these attributes is acknowledged and also confirmed in the data.

For the purposes of answering the research questions, the results focus on the mainly with the 'hard' skills linked to CS. The deductive 'hard' skill codes informed by the CS 'experts' proposed computer science links to process thinking. The thematic analysis shows process thinking is a valuable skill for non-digital industries.

3.2.1 Process related themes

The four process related themes arising from the interviews are as follows and defined in Table 5:

- **Theme 1:** awareness of processes around them and the business problems that they solve
- **Theme 2:** information extraction and knowledge representation (model building)
- **Theme 3:** improvement via creation/adjustment of processes and
- **Theme 4:** expressing processes verbally or in writing

Theme	Definition	Occurrence
Awareness of processes around them and the business problems that they solve	Employee’s ability to understand the processes that directly and indirectly relate to them. This includes awareness of processes to fulfil their own role and those in the external environment and may affect productivity, supply and demand.	100%
Information extraction and modelling	Employee’s ability to extract key and essential detail specific to the task. This may include relevant information from complex processes and creating models.	100%
Improvement via the creation or adjustment of processes	Understanding the business sufficiently well to achieve a desired goal through process improvements or creating new ones. This skill may be necessary if an issue occurs in the process creating a problem that needs solved. An employee may also suggest better ways of working because they are immersed in the process they can see where value can be added.	88%
Expressing processes verbally or written.	Companies describe the importance of employee’s ability to share practice in different forms such as verbally or in writing. Employee’s active engagement in understanding and articulating process descriptions enables the sharing of practice to improve further processes across the company.	88%

Table 5 Themes arising from interviews with non-digital industries.

3.3 Interview themes

Each of the CS oriented themes arising from the interviews will now be discussed in turn.

3.3.1 Theme 1: Awareness of processes around them and the business problems that they solve

All interviewees discussed the need for employees to be aware of the processes relating directly to the role that they fulfil within the company and the ability to see beyond this to processes in the external environment that may affect supply and demand. This external influence could be related directly to clients, or in the wider world.

For example, the confectionery distributor commented on the retailing cycle in a particular store, identifying the importance for a salesperson to be aware of that store's process, and in general to be alive to changing processes among their clients:

‘that store has a promotional cycle, so we would look to call at the right times of when the retailer would implement the promotion. So if the promotion goes live next week, and you call this week, you're calling at the wrong time, if you call next week that's an effective call. ... As you grow through the business then it's important that the individual is able to not be spoon fed, but they research those trends themselves.... having that mental agility that they don't just stick to the same things that they do again and again and again.’

Participant A

The photographer describes how he must align his process with that of the printing company he occasionally works for:

‘They have a design team. Basically, I get given samples. I've been doing it that way, I know how they want everything shot, so I do it and I send them different examples and stuff like that. They're using other things to put a catalogue together, so I need to make sure that the images are of the correct resolution and different things like that. You definitely need to work within perimeters as well.’

Participant B

More generally, he discussed how he had to consider this aspect with the majority of the clients for whom he worked. The energy company manager described at a higher level

how the leaders of larger work programmes had to ensure that potential interactions with other programmes were understood:

‘every major project has programmes, and those programmes need to be delivered by the people on the ground, and the people who are organising those programmes need to understand what is required of them and how they interact and interface with all the other people.’

Participant C

The joiner/decorator noted that other trades, ‘like the electrician, the plumber and the heating engineer all have their own processes as well, so you’ve got to marry yours with theirs”. When asked whether the apprentices he was training needed to understand the larger process, he replied:

‘Completely, because if they don’t, like in the line of work that I’m in, if something’s not right from the start, it’s from the foundations up, if we do something wrong, if you’re not going to make a wall right, the next person who’s going to put on the skirting boards, so that’s not right, you’re affecting later trades.

Participant D

The biscuit company manager commented on the problems when this context is poorly understood

3.3.2 Theme 2: Information extraction and model building

Many interviewees described activities of their successful employees that are based around the selection of specific information from complex contexts. While this is not a direct component of process-oriented thinking, it is closely related to computational thinking – one a high-level definition of which could be the study of processes that manipulate information.

Participant D

Describes an employee who built models of the production of the biscuit company was described as follows:

‘He’s taking loads of information from various departments because, obviously, the things that he’s looking at, it’s inputs into bakery and outputs for the bakery into the packing and then packing, there are various, depending on the process, it goes through what variety it is and different so it depends on the variety, it depends on the line it’s on, it depends on what product it’s going into. So, there are so many variables in there.’

Participant E

Confectionery company employees require to have high quality listening skills to pick up from the client the detail of how their shop works – storage of goods, footfall, amount of time spent in the shop.

‘One of the key things they have to identify is a customer’s needs. They have to have the questioning and listening skills, rather than just taking out the head office instruction and taking it into any retailer; they have to make it relevant for the retailer.’

Participant A

The interviewee for the energy company highly rated the following aspects of information capture and modelling:

‘Understanding a situation, drawing it all together, prioritizing solutions, mapping things out, being able to look forward. But also, they have to look backwards and

say, if my end product looks like X, how do I get to that point? What do I need to do to get to that point? And then working backwards and saying, Well, actually, if that's the end point and it's in six months' time, I actually needed to start three weeks ago, and then recognizing you already have, perhaps, an issue that needs to be solved, that either the end point needs to be pushed out or more resources required in the interim, and what have you.'

Participant C

The joiner stressed the importance of seeing both the end result and the steps to get there in his / his apprentices' mind's eye:

'Look at it and start picturing it in your mind what you're going to do, where you're going to screw a bit of wood, where you're going to cut that.'

3.3.3 Theme 3: Improvement via creation/adjustment of processes

Companies describe various mechanisms for the sharing of practice to improve processes ranging from those bespoke to the organisation to more formal continuous process improvement approaches such as Lean Management Philosophy (Brioso 2015). Employees engagement within these approaches requires an ability for extracting key and relevant information. In addition, interviewees described some less formal improvement for their organisation based on the adjustment or creation of processes, prompted by their own or their employees' insight.

Participant A

Commenting on a particularly successful project outcome, the photographer said that he succeeded because:

‘I physically manage to get everything to work and go at the same time always improving what I do’.

Problem solving was mentioned often, in the sense of producing a one-off process to produce the required result.

Participant G

The Tissue Solutions company, transporting delicate human tissue around the world in a timely manner is the key company activity and a key quality in employees is developing creative alternatives in a very tight time frame for delivering the tissues when the agreed mechanism had failed.

Participant H

Speaking about successful employees, the energy company said:

‘Any individual who’s bright and attentive is an asset. If they are sparked by ideas and pulling ideas together and making sense of information, understanding processes, You can’t do B before you’ve done A. I guess being logical, those kinds of skills are of great importance, and that’s not really going to matter where they are within the business, the logical process-driven individuals.’

He noted additionally that reasoning about processes is important, in particular:

‘An understanding of cost effectiveness, solution A is brilliant, but it’s really expensive. Solution B isn’t quite as good but it’s a fraction of the cost and therefore is easier to implement, or to justify or what have you”. So it’s people who can balance, and that probably comes down to the reasoning thing, taking information, looking at the process and the model and saying, Well, there are three options you can do here. One is the quickest, one is the most expensive, and this third one is a hybrid of all of those, and therefore it’s probably the one that’s best.’

Participant D

In the biscuit company, the employees who really stand out are the ones that basically, don’t sit back. They are not afraid to bring forward ideas and suggestions and are very hands-on. They are not just senior people either there are wee stars all over the place in various departments. Updated processes that had increased productivity by 20% and reduced the level of waste being sent to landfill by 80% were initiated by staff:

‘Seeing the job and seeing where they can add value into that process, whatever it might be doing. A lot of the jobs are not highly skilled or anything like that but you still can make a difference in the area that you’re in. We’ve got one of the boys in the bakery department, he’s went on to one of the lean teams to look at things. He’s been looking at the bake profile of the biscuits. So they’re taking certain ones and what they’re trying to do is increase the output but over the same – the same number of mixes will go in but what can they do in the bit in between that and the packing to get more biscuits through in the time that they’ve got? So they’ve been looking at bake profiles and different ways of getting either more biscuits in a row or whatever, or speeding up the bake or the cooling time...’

Participant E

Exploring processes to determine where waste can be reduced seems to be very profitable. The confectionery distributor pointed out that in some of his plants made real financial gains by individuals working out how things could be improved.

‘Just simple ideas to remove wastage have saved the company millions.’

Employees are measured on their ability to organize their “territories”. Although the size of the territory and the required number of contacts is generated from the centre of the

company, the way they go about those calls depends a lot on local knowledge. The interviewee pointed out that

‘an ineffective employee would be driving all over the place, going from one down to another and then revisiting that town three days later. An effective person will plan their journey for 19 days. They work in 19 day cycles, so they would plan for the 19 days.’

Participant I

The drinks company saleswoman understood the product placement issues and timeliness of a successful sales intervention. About their client:

‘They were dual stocking but pushing the other brand rather than ours and they basically were selling 20:1 of the other brand so I cold called spoke to whoever I needed to speak to and we had follow up conversations because we wanted to win back the volume we lost. Basically, they were saying the deal we offered wasn’t good enough for them so I kind of left it and we offered a massive support over freshers week anyway I left them to it. I got in touch and they had actually removed the second brand from their bars or three of the four... and had sold more volume in the freshers two week period than they had in the previous year.

Participant D

The confectionery retailer said:

‘The way we do it is something that is just very similar to either Twitter or Facebook, it’s an internal thing that they have on their phones. So they see something in store for the first time or have a success in store, they can actually post that on this site within minutes.’

Furthermore, the retailer added that:

‘What we look to encourage is that the improving of processes isn’t just a leadership role, and we often run focus groups with our front line reps to get their views on how things could improve, because actually the people who are probably best placed to improve the processes are the ones that do it daily.’

3.3.4 Theme 4 Expressing processes verbally or in writing

Employees are encouraged or required to express process descriptions in various forms. Companies described various mechanisms for increasing the sharing of practice to improve processes.

Participant J

The hospital construction team noted that:

‘We took people from the site operatives, supervisors, managers and the health and safety manager’s team so it’s a mixed team and mixed skills set and they are all able to report on what they don’t think is quite right and able to discuss that in an open forum and then re-conceived the challenges, deciding if something needs to be addressed and how they will address it.’

Participant K

The energy/windfarm company operate a License 2 Innovate process for employees to identify and articulate potential improvements:

‘So they may have said, “I don’t know why we do it like this, why don’t we do it like that?” They would write a paper, that goes through – the various parts of the business have their own innovations committees and people look at it.

Fundamentally, what somebody is trying to say is, “We could be doing some faster, we could be doing something –“ when I say, “Cheaper,” we are not talking about cutting corners, we’re talking about not wasting money, not wasting time, not wasting people’s time, not wasting resources. Because actually, if you could use half as much paper or drive half as many miles, let’s do those things. There will have proposals that may have saved £3.50, and there may be proposals that have saved £35m, and all of them are valid and welcome.’

Participant H

The energy company must follow strict industry guidelines on design and processes and these are usually captured in a spreadsheet.

Participant I

The pump manufacturer noted that:

‘after requirements were agreed with a client, the detail had to be transferred to the engineering team for a more detailed design – this could be sketched or with a drawing package on a computer.

Companies describe the importance of employees ability to share practice in different forms such as verbally or in writing. Employees active engagement in understanding and articulating process descriptions enables the sharing of practice to improve further processes across the company.

Participant M

The logistics manager describes the importance of knowing about process and how they connect with business productivity.

‘I’ve had people that know the information and know the process but no understanding of the relationships and how it fits in they’re in their wee silos, “Oh, this is my job; I know my job really, really well. I do it really, really good”, but there’s no, “I don’t know where this comes from and I don’t know where that’s going”. I think they need to understand that because anything goes wrong; how do you know where to look?’

Participant I

The drinks saleswoman described one route to share information:

‘If I want to do something good in an account, I make a presentation to the boss. Obviously, he doesn’t know my area the way I do so I have to let him in on my area I have to create a picture so that he can decide whether or not he thinks it’s a good idea to work with certain accounts and do certain things with him.’

Participant E

The confectionery manager spoke also about changes in larger societal activities and how it was essential that they kept abreast with these if they were to effectively advise their retailing clients on the best way to market the confectionery:

‘More and more, because of lots of different lifestyle changes, people are shopping three or four times a week, and going to local shops. So some of the lifestyle

changes for that might be the price of petrol, the lack of space in your house to buy a big shop, the cost of shopping, so it's easier to buy smaller purchases than one big shop.'

3.4 Discussion

3.4.1 Process thinking and computer science

The interviews show clearly the value placed by employers on process thinking (PT). This section explores the relationship of the PT valued by non-digital industries with computer science content.

Guzdial et al. (2019) states that computer science is 'the science of processes; all processes.' A subset of these processes is primarily algorithmic in nature, but to deal with the large range that computation can model, it is much more appropriate to 'think all systems' and to see the representational possibilities of the computer.

The following section shows the relationship of each of the emerging process-oriented themes from the thematic analysis. It provides a view of how Process thinking (PT) is of value to all school children irrespective of their chosen career, and that it lays the foundations of the CS concepts for those specialising in a computer science pathway. It claims that PT is a precursor to learning CS concepts.

3.4.2 Theme 1: Awareness of processes around them and the business problems that they solve.

Employers value employees who first understand the processes that they execute and how this relates to the multilevel processes that directly and indirectly relate to them in the workplace. This type of thinking is a pre-requisite where problems are conceptualised in understanding before solutions can be formed. PT in this context, is an easier concept to grasp and teach than computer programming concepts such as algorithmic thinking, since processes can involve physical, tangible objects and humans; and yet it applies more widely. This is important because there is a very distinct difference between the broad process thinking described in the study and the accurate and precise rules of a CS algorithm. Defining algorithms is not in scope of this study, however, the notion of algorithms as a series of steps applied in a real-life context feature in some primary school unplugged activities. The loose definitions used during unplugged activities can blur the distinction between the strict characteristics of an algorithm, however, they do align with the more relaxed qualities of processes generally. By explicitly talking about processes within the context of algorithms, the qualities of algorithms can be highlighted avoiding misconceptions.

3.4.3 Theme 2 Processes for continuous improvement

All participants view employee engagement for their organisational improvement as important. Their view that effective employees are change catalysts creating new business models (Mitchelmore and Rowley, 2010) is supported in literature. Structures to support organisational development mentioned by participants varied from informal ongoing discussion to formal system improvement techniques such as the Lean Management (Klein, L.L., Vieira, K.M., et al., 2022) model and Six Sigma (Karakhan, A., 2017) analysis with its Five Whys Technique (Serrat, O., 2017). These formal structures use tools to facilitate discussion and improve employee participation to achieve ‘perfect, sustainable workflow while minimising waste and being readily adaptable to change’ (Fatimah, Y.A., Govindan, K., et al., 2020).

3.4.4 Theme 3 Information extraction and knowledge representation

The formal and informal improvement processes also require employees to understand the modelling ‘tools’ that the processes use to elicit information before they can build or create a solution. Modelling in CS is, in these cases, creating visual representations by extracting relevant detail of the information processing and understand how they apply to the modelling tools. This process of abstraction requires the ability to ignore unimportant information relating to the processes that they are involved and in some cases level prioritising processes that cannot be ignored.

3.4.5 Theme 4 Process thinking for problem solving

This theme builds upon the three mentioned above. Employees who understand information and process and the tools that manipulate these are well placed to create and evaluate solutions or problem solve. This problem solving can be part of the high-level organisational approach or less formally on an ongoing basis. Problem solving more broadly involves, employees breaking down problems step by step understanding the multi processes at play. They extract information often creating visual representations of what is needed and make suggestions or improvements for themselves and others to follow. In CS this is known as reasoning and PT corresponds to a crucial level of reasoning in designing and reviewing computer programs. Reasoning within the context is where mental, written or computational models are created using tools that a computer can execute. Reasoning is complex, and so PT problem solving would be a useful precursor to learning that later.

It is worthwhile at this point linking the findings to the research focus on the value of CSED for young children in primary school formal education. Background reading shows that CS curricula for young children consists of programming. The next section aims to show a possible relationship for CS, PT and the programming experience children are likely to encounter in their primary education.

CS can be described as the subset relationship of Computer Science - Programming - Process Thinking (CS-PR-PT). Papert (1980) refers to the relationship between programming and thinking skills. He argued that using Logo for programming could facilitate children's procedural thinking. His view is computer science and in turn programming which leads to a type of thinking of value across a range of contexts. Arguably, underlying programming skills could therefore be of value to all. Emerging CS practice in primary school follows this rational with children learning about algorithms, abstraction, decomposition, and control structures. This thesis suggests that curricula are typically set up with CS leading to PR leading to the PT relationship (CS-PR-PT). Reflecting on the data gathered from the study, the findings imply an alternative view that a better introduction to CS in the primary school could be PT-PR-CS. This approach enables the more precise core CT concepts to be built upon through a progressive approach. In addition to the benefits for all, it may be a better introduction to programming because it can sidestep a central problem; that of learning by finding code online or writing it by shallow trial and error (Wong-Aitken, D. Cukieman, D., et al 2022) without understanding the process and clear strategy on self-reflection (Schulte), and it can solve the set problem, but without a student understanding it (Wong-Aitken, D. Cukieman, D. et al 2022). This skill can be compared to how useful it would be writing essays by cutting and pasting from online sources and never learning to compose English text for yourself.

Computing Science (CS) academics, school sector educationalists, and digital industry employers believe that some kind of CS education in schools is desirable, given their direct contribution to the development of K-12 CS curricula. Either CS-PR-PT or PT-PR-CS, this study implies a clear rationale for teaching PT to non-specialist CS learners at an early stage in their learning. It also identifies what primary teachers should consider from the large set of concepts related in different ways to digital applications.

If it is accepted that process thinking (PT) aligns with key CS programming concepts, then this paper has made a principled case for including it in the early years of education. It has arguably, demonstrated a value for PT that spans right across non-digital industries, whether computer based or not, as a key attribute of many companies' most successful

employees. Teaching PT in primary education could form a simple yet powerful underpinning conceptual framework that prepares all for possible later extensions to learning more than computing, while at the same time ensuring that they have a fundamental thinking skill for almost any future career. Indeed, the authors' home country is considering defining its primary and early secondary CS curriculum for schools according to this framework.

The skills that Microsoft, Facebook and Google feel are valuable as industry leaders, may not be the same skills that are valued by the small businesses that drive most advanced economies around the world. These voices have not necessarily been heard by those devising CS curricula for compulsory education. Industry is important and most educators will concede that listening to those that drive the economy activity is important when devising effective compulsory curricula for all. Industry can influence significant changes in society and support many individuals to learn new skills. However, the voices that are often listened to from industry regarding computing science are technology companies, typically large multinationals that span much of the developed world.

Finally, PT can be truly recognized as an enduring educational goal, enhancing opportunity for all, since all stand to gain from it, as they do from literacy and numeracy.

Teaching PT in primary education can then be argued to prepare all for possible later extensions to learning more computing, while at the same time ensuring that they have a fundamental thinking skill for almost any future career.

3.5 Future Work

This study gives some detail about what one sector of the job market requires and starts exploring how these skills of value relate to computing. Therefore, with the expansion of the subject to younger learners, there is a need for further insights on what they really need to learn and importantly how they are taught. With the introduction of CS in primary schools started, there is a need to understand typical approaches and ensure that learners are indeed developing those much-valued skills for all.

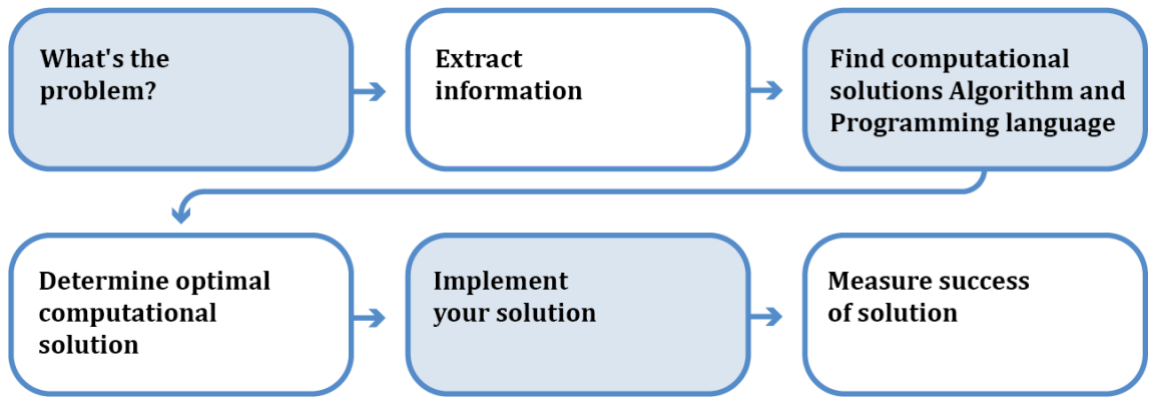


Figure 2 Process Thinking strategies in a primary classroom.

Quadrant B: Typical approaches to introductory programming in primary school classrooms

Quadrant B is set in 2016 at the time when introductory programming is beginning to feature in primary school classrooms. It is informed by quadrant A conclusion that, process thinking, an output of programming is valued by non-digital industry employers. Given CSED is of greater value for younger children in their formal education, the thesis focuses on exploring if and how it's being implemented within the primary school classroom. Quadrant B is therefore organised into two chapters. The opening chapter explores related work on CSED in primary schools. The thesis focus is to establish typical approaches and therefore there is a focus on IP pedagogy. Literature on introductory programming pedagogy in authentic classroom situations is patchy. This gap of insights from literature on introductory programming leads to an exploratory study in 4 Scottish primary schools. Findings show all educators in the studies create scripts first using step by step instructions in block programming environments. The course content is led by the resource itself and importantly, a few children are demotivated by the experience.

RQ 2: What typical programming approaches are in place in traditional primary classroom and how successful are they?

Chapter 4

4.1 Quadrant B

Summary

The evidence base on CS approaches in authentic primary school classroom situations is limited. Most studies focusing on theoretical content provide valuable insights on commendable actions by the CSED research community to support system wide change. However, there is a gap in knowledge about practical applications or hypothesis testing in classroom conditions. The purpose of this study is to add to the existing evidence base and explore the implementation of an introductory programming course in an authentic classroom learning environment. A mixed methods approach with electronic survey led to an exploratory observation of teacher's use of resources and children's outcomes in an introductory programming course. The study took place in 4 schools with 4 educators and 97 children. Teachers planned and delivered learning, findings show that teachers choose visual programming environments Kodu and Lego WeDo. In two schools, the teachers recruited the same IT instructor with no teaching experience to deliver the lessons. In two schools the teachers planned and delivered the learning themselves.

4.2 Introduction

At an unprecedented rate, many countries are introducing computer science (CS) into their K-12 curricula and although CS is neither programming nor computer literacy (ACM Inroads 2, 2011), literature does suggest that most Primary school CS content features programming (Mensan, Osman et al., 2020.) in the form of tasks related to computers (plugged) and off computer tasks (unplugged). Programming issues in tertiary settings are well-documented. Issues for novice programmers include problem solving, motivation, engagement, learning syntax and a lack of appropriate methods and tools (Medeiros, Ramalho, et al, 2019) and its introduction to K-12 is complex, requires systemic change, teacher engagement, and development of significant resources .

The CSED research community rose to the challenge of providing insights on introducing mandatory CS in schools. In 2011, an article published in the ACM inroads recognised their role in the CSED K-12 change process. The article set out a clear vision to support curriculum implementation (ACM Inroads 2, 1 (March 2011), 48–54) in relation to computational thinking (CT), which they state, has an important relationship with the CS practice of programming (National Research Council, 2010) (Denning, 2003).

Collective efforts by the CSED research community led to the development of resources and the publication of studies based on theoretical content and some data driven experiments. This study reviews literature on approaches and identifies trends and patterns in introductory programming tools and instructional methods. It looks briefly at the theoretical content literature and focuses on publications that test out introductory programming theory within the primary school context. Findings highlight a gap in studies set in typical primary school classrooms and that all data driven studies in classroom settings are supported by high adult student ratio led by the CS researcher. Studies involving children show variations in success rates. The related work section concludes that the studies, due to their set up are not replicable in typical classroom settings. Therefore, given the significant challenge of training sufficient teachers (Brown, et al, 2014) in line with the implementation rates of CS in primary school curricula there is a need to know how primary teachers without training are implementing the subject. This insight will support future work and resource development.

The study has one RQ:

RQ2: What typical programming approaches are in place in traditional primary classrooms and how successful are they?

To answer the RQ a review of related work was undertaken and identified a gap in understanding what primary teachers do with the available resources when left to plan and implement CS on their own. The pilot phase involved an electronic survey issued via Twitter, followed by an exploratory study in 4 Scottish primary schools (schools A, B, C and D). The survey in the study showed that most primary school introductory programming was taught by secondary teachers, visual programming tools such as Scratch were used as the resource curriculum. In the follow up exploratory study, school A,B,C used Kodu visual programming and school D used Lego WeDo . School A and B were taught by a local authority IT employee with no primary teaching experience. Schools C and D were taught by their class teachers. Instructional design included commercial resources driving learning and step by step instructions. In all schools, children's success rates varied and in one school, teachers were unclear of CS concepts that they were teaching.

4.3 Related work

This study draws on the work of Szabo, Sheard et al (2019) who in a systematic literature review of K-12 introductory programming 2005 to 2017 recognise that “despite a plethora of work focusing on the definition and implementation of a K-12 curriculum that covers computational thinking, computing or digital technologies more broadly few works look at introductory programming”. The systematic review of the K-12 introductory programming literature from 2003 to 2017 and identified K-12 introductory programming papers and those other aspects of introductory programming such as CT where this intersect with introductory programming. The original themes from Szabo et al (2019) provide a starting point for the initial analysis. The thesis focuses on K-6 (elementary or primary school stages of formal education) Further analysis of 20 papers from the original 72 resulted in two categories. Theoretical driven and stat driven studies. The next analysis established ‘topics that occur’ and re occur” (Bogdan and Taylor 1975:83) or are recurring regularities (Guba 1978:53)

Szabo, Sheard et al (2019) applied their search to a previous systematic review on introductory programming literature. ACM, IEEE Xplore, ScienceDirect, Springer Link and Scopus. Search terms include “introductory programming” OR introduction to programming” OR “novice programming” OR “novice programmers” OR “CS1” OR “learn programming” OR “learning to program” OR “teach programming” resulting in 108 papers in K-12 education.

For this related work section, the 108 papers were reduced to 20 when individually examined for K-6 (elementary or primary school) context. Each of the 20 papers from the review relating to K-6 were read and analysed individually. Two main types of studies emerged, those with theoretical content and those that are data driven with few providing samples that represent a larger population. The theoretical content studies describe a call for the CSED research community to support change in the education system. The data driven studies aim to test out theories in classroom settings, they are exploratory or show the effects of interventions.

At this stage, repetition of terms supported the identification of themes. Some of topics that reoccurred (Bogdan and Taylor 1975:83) with recurring regularity (Guba 1978:53) (Ryan and Bernard 2003) resulted in *curriculum*, *programming tools*, *materials created by academics* and *instructional design emerging from the studies* providing the organisers for the related work section. Each of the organisers contain theoretical content and where appropriate supported by studies involving school children. The related work section concludes with an overview of the theoretical content studies and the exploratory data

driven studies highlighting a gap in literature at this time of insights into introductory programming in a traditional primary school classroom.

4.4 Curriculum

The primary school curriculum typically consists of a set of expected subjects or standards (Priestley and Minty, 2013). In relation to primary school computing science, curricula are being developed with content to equip children and young people with the skills for life, learning and work (Rose, 2009). K-12 CS curricula may include how computers and computer systems work and design and build programs (Sentence and Csizmadia, 2017). Adding computer science as a separate school subject to the core primary curriculum is a complex issue with educational challenges in relation to the design of the curriculum, and the knowledge teachers need to teach the curriculum (Charoula et al., 2016).

Primary school CS curricula typically include programming and the developmental milestones of a young child add an additional level of complexity for consideration. The view that young children are unable to abstract is now being challenged when developmentally appropriate approaches are in place (Waite, Curzon et al, 2016). Armoni, M., (2012) states that early exposure during kindergarten is necessary and found that young children can think abstractly when concrete reference systems are used to situate their thinking (Angeli, Voogt et al., 2016). In order to support the curriculum various programming tools were developed by the CSED research community to support system change and a number of fieldwork studies take place involving children using these materials to learn CS concepts and address the developmental needs of young children.

4.5 Programming Tools

Computing lessons in schools could be an effective learning experience through the range of introductory programming tools available. Visual programming environments such as Scratch and Kodu, plus tangible objects are freely available for primary school teachers to lift and use. However, choosing between the various programming tools, activities and instructional methods is quite challenging. Both experienced and novice programmers use examples while programming, whether from tutorials, forums, or source code. Author's Ichinco, M and Kelleher, C., (2015) ran an exploratory study to understand the hurdles encountered and strategies used by programmers working with examples. They recruited 21 children aged 10- 15 from the St. Academy of Science mailing list. While the age range of the study is outwith primary school age it is worth exploring. The author's conducted their study using Looking Glass, a drag and drop novice programming environment where

the output of a program is a 3-D animation. They created six completion programs based on six concepts of varying difficulty. The instructions for each task asked participants to add to or modify the given program to create a specific animation. To analyse programming behaviours, a 'realization point' is defined at the time when the participants discover the crucial concept in an example. Results show that participants spent more time after the realisation point using the example than they did identifying which part of the example to use. The study concludes that of the 54 tasks participants completed, 37 (69%). The time taken to complete tasks ranges from 1.63 minutes to 8 minutes. These results shows variability in the progress of children participating in the same activity.

It is noted that, popular interest in robotics has increased in the last few years. Robotics is seen by many as offering major new benefits in education at all levels (Benitti, 2012).

Benitti's (2012) systematic literature review searched IEEE XPLORE, ACM Digital Library, Science Direct, Springer Link, ERIC Educational Resources Info Centre, Wilson Education peer reviewed articles written in English and published between 2000-2009 online bibliographic databases returned, ten relevant articles in the study of which 5 relate to primary school age children age 5 to 12. They report that educational robotics usually acts as an element that enhances learning, however, this is not always the case, as there are studies that have reported situations in which there was no improvement in learning. Text based programming tools such as Logo are promoted; this mini language was explicitly developed for teaching programming. Logo permits the user to focus on a much-reduced set of instructions with an adequate, clear syntax. Each of the words in this vocabulary corresponds to one instruction the turtle can unambiguously understand and execute. Simple Logo editors do not rely on a click-and-drag approach and allow the pupils to type the instructions by themselves (Serafini, 2011).

One study in Switzerland describes researchers actively involved in reaching out to primary teachers and training them to successfully teach programming and computer science through Logo. Teachers have no prior CS knowledge and experience the materials created by researchers in advance of the children. Each lesson is supported by a lecturer who takes responsibility for the class and learning, a CS assistant and the class teacher supports the class but not actively involved except for teaching one concept under the direction of the lecturer. The whole approach is didactic, and children follow a booklet containing simplified instructions and work autonomously. Although the study takes place in a classroom, with children aged 10 to 12 in 5th and 6th grade, the authors do not report data or experimental conditions (Hromkovi, et al., 2016). In addition, the children are taught by the lecturer and there are 3 adults in the classroom.

In the field of introductory programming, many tangible systems have been designed for children having as their purpose to connect to the programming activity with the physical world. Another study (Sapounidis and Demetriadis, 2013) led by academics in a Greek school involved 61 children age 5 – 12 measured children’s enjoyment and ease of use using graphical and tangible interfaces. Statistically significant results show that younger children find tangible interfaces easier to use. Older children preferred graphical interfaces. Authors upscaled the work in a cross age look at the performance of 109 children age 6 – 12 in one tangible and one graphical system. Once again, children were taught by researcher and volunteers on the project were randomly assigned. However, it is not clear from the study if it took place in a typical classroom setting or if there is variability in children’s progress.

4.6 Materials created by academics

As mentioned previously, the call for computer scientists to support CS K-12 provides teachers with multiple options for teaching introductory programming. As a result, computer scientists developed a variety of educational programming platforms to engage young children in computer science activities. For example, researchers in Sweden (Serafini, 2011) show academic led initiatives programming resources to develop CS for primary schools across four of their main geographical areas. They conclude that computational thinking can be developed through programming. Two examples of implementation approaches in place include Bebras as a motivating and playful online resource for young children and Lund university engaging in activities related to computing at school through its science centre programming for everyone (PfE) project. Two aims of the PfE are to develop teacher training so that pre-university schools for all ages can help young learners to discover the excitement and importance of computer programming (Heintz, Mannila et al., 2015). They conclude that further work is required to bring about improvements.

Scratch visual programming environment is a popular choice for introductory programming. A review of Scratch, Scratch Jr and Blockly visual programming tools in one study identified a gap in the usability for 4-6th grade children. In response to the gap (Hill, Dwyer et al 2015) designed a programming environment ‘La Playa’ which would support elementary teachers with or without CS backgrounds and overcome some of the practical challenges faced in a traditional classroom setting. They piloted a curriculum based on Scratch on 4th to 6th graders in 15 Californian schools. Each class was supported by 2 tutors from the research team and the class teacher. The analysis focused on two

projects and findings show that the Scratch interface features, and math content were too advanced for 4 – 6 grade and ScratchJr which was developed to address the issues of subject knowledge alignment, is claimed by the authors as not developmentally appropriate for 4 – 6 grade. It is noted that 60% of children achieve 6 out of the 9 tasks possible, 47% achieve 7 tasks and 3 children solved 9 tasks which authors report this as ‘remarkably good’. Looking at this outcome from a different lens, arguably 40% of children did not achieve 6 tasks. While this study is of interest and provides some insights into visual programming issues, there are the following limitations to the work, The schools involved had varying numbers of classrooms, grades participating, start dates, and order of projects. In addition, this study does not report on methodology, measures or use quantitative data. Further examples of CS created resources are found in Argentina’s nation-wide computer science at school initiative. Researchers Sanzo, Schapachnik et al (2017) introduce Pilas Bloques, a scenario-based children’s learning platform. Their strategy based on inquiry-based learning through short and focused challenges and abstraction building takes account of previous research by universities. In a previous study not available for analysis, the platform was tested on students age 9 to 15 resulting in a teaching sequence and a manual which is justified. It does not mention the study being undertaken in a traditional classroom setting. The authors conclude the need to trial the materials using experimental conditions on same age school children.

Recognising children’s developmental milestones as an issue that needs addressed when creating introductory programming resources, a study was undertaken using Papert’s constructionist theoretical framework (Bers, Flannery, Kazakoff et al 2014). The framework states that children can learn deeply when they build their own meaningful projects in a community of learners and reflect carefully on the process. It underpinned the curriculum developed by researchers given to children in a kindergarten. The curriculum itself, takes approximately 20 hours of classroom work, includes six structured 60- to 90-min activities culminating in an interdisciplinary project. All the activities focus on young children building and programming a robotic vehicle to accomplish a particular goal. Participants have a mean age of 5.6 attending kindergarten in the Greater Boston area. The teacher delivers the lesson directed by the researcher, 20 researchers support the work. Each lesson addresses one or more powerful ideas. The study using robotics demonstrated that kindergartners were both interested and able to learn many aspects of programming. Quantitative data drives the findings for the quasi experiment. Outcomes are variable with 53 out of a possible 63 children completing the activities. In the first three activities 75% of children achieve target level. There is a trend of decreasing achievement in scores across Lessons 3–6 where more sophisticated concepts were introduced.

Researchers describe concrete experiences in teaching programming in Logo at Swiss primary schools, using didactic visions. They consider prospects for long-term empirical research. In Switzerland, academics (Serafini, 2011) developed ad-hoc teaching materials for primary schools. Their study involves 6 lessons for children with no prior knowledge in programming implemented in 4 different projects. The study focuses on an analysis of project one and project two. Project one involves 12 children, their class teacher and 2 tutors from the research team. Children work autonomously through materials to complete programming exercises and pace differences emerge by lesson 4. Project 2, 2 classes with 31 pupils, each class supported by their class teacher and 2 tutors from the research team. No measures reported in children's progress except for 2 females achieving full marks

4.7 Instructional methods

Learning to program is generally considered to be a complex cognitive activity resulting in a high intrinsic cognitive load. Having exemplified some of the resources created to support implementation, the related work section moves to the instructional methods. The materials created in one study (Hromkovič, Serafini, et al., 2017) are freely available online and aim to reduce cognitive load through reducing the impact of factual knowledge, focus on one looping control structure that avoids variables and use didactic design principles. In their experiment they measured cognitive load experienced by pupils and teachers while programming in Logo, compared the materials with predecessors in terms of usability and benchmarked three programs. The aim was to apply the intervention in a typical classroom. They introduced 75 children and 12 teachers to programming in Logo in groups of 12. The experiment confirmed that most children experience rather high cognitive load while programming, which reinforces the need for tailored teaching material to enable better learning experiences. In a further experiment the researchers delivered revised materials to 18 5th grade pupils with 85% meeting expectations. The study claims that they ran classroom studies and found a clear need for didactic settings. It is not clear the role of the class teacher in the experiment and the class teacher's involvement was not reported upon.

According to Harms, Balzuweit, et al (2016) children's programming environments predominantly use two instructional formats to support learning: tutorials that present step-by-step instructions for learners to follow and code puzzles that provide learners with a set of code elements and a specific challenge. Their study took place in The Academy of Science which is a not-for-profit organisation in the St. Louis metropolitan area dedicated

to science outreach. They explored 30 learners between 10 and 15 with a mean age of 11.2 perception of value in each approach. This study took account of participant's prior experience. Findings show participants cited length of task as a source of ease or difficulty and they spent significantly longer on puzzles than tutorials. Overall, 83% of participants explained their perception of difficulty through their own experience at least once during the study. There is no indication that this work will be replicated in a traditional classroom environment. At the time of writing UMC (Lee, Martin et al.,2011).and PRIMM (Sentence and Waite, 2017) studies involved children in secondary school education and will be covered later in the thesis.

Finally, children collaborating on learning is an expected norm for teachers when planning learning in primary schools. A study (Lin et al 2007) in Taiwan involving 3 classes reviewed guided collaboration on ninety-four sixth graders. Each class is randomly assigned to an individual learning group, a free collaboration group and a guided group and authors investigate their achievements learning to program in MSWLogo. Standardised pre-test measures and verbal intelligence tests were applied and there was no significant differences between each of the classes involved. Each of the 80 lessons across 14 weeks are led by an instructor in a computer lab. A statistical analysis of students' test scores showed that the guided-collaboration group significantly outperformed the other two groups. The differences among the low achievers of the three groups were especially significant, indicating that they had benefited the most from guided collaboration. It was also found that guided collaboration promoted meaningful discussions among students, which had contributed to their better understanding of programming concepts and problem-solving skills, and hence better test scores.

Conclusion

Many of the introductory programming resources described above have been evaluated and show promise in a laboratory or modified classroom settings. However, the 2016 evidence base on CSED research community publications in a traditional primary school classrooms is patchy. As mentioned, previously, there is an acknowledgement that other works exist such as Waite (2016 and 2017)⁴ and (Sentence & Waite, 2018). However, this background

reading was based on Szabo, Shead et al (2019) K-12 IP systematic literature review. The studies discussed in this chapter are distinguishable by two perspectives namely, theoretical content and data driven. Theoretical content does not depend upon an experiment and is based on developing theories. Data driven studies on the other hand, can be classed as non-experimental or experimental. Non-experimental studies have no control group or other comparison to assess participants abilities. Experimental or quasi-experimental are those that include rigorous comparison with a control group.

While theoretical content and data driven studies are both important, this study was interested in understanding traditional classroom practice in a primary school introductory programming course. However, few studies involve a control group, all are led by the researcher with a high adult to child ratio. These variables are not typical of a primary classroom setting. As a result, little is known about how teachers deliver introductory programming, independently in a traditional classroom setting. The table below provides an overview of the Szabo, Shead et al (2019) systematic literature review studies used within the related work section. Column 1 presents an overview of authors, country of study. In column 2, there is a brief description of each studies goal. Columns 3 shows the methodology as either non-experimental (X), quasi-experimental(√) or the experiment cannot be categorised due to insufficient information available (∅). Columns 4 – 11 show; (4) the number of participants; (5) minimum age of participants;(6) Variability in progress of participants; (7) Control group; (8) Teacher led; (9) Researcher led; (10) Study took place in a physical classroom setting; and, (11) The number of adults supporting the group of participants. Arguably there is no confidence that these studies are replicable in a classroom or able to be upscaled to test out the validity. While it is recognised there is a lack of adequate empirical evidence in terms of the effectiveness of the frameworks proposed herein, it is expected that our knowledge and research base will dramatically increase, as more countries around the world add computer science as a separate school subject to their K-6 curriculum

<i>Author(s) and Country</i>	<i>Study outline</i>	<i>methodology *</i>	<i>Participants</i>	<i>Age (minimum)</i>	<i>Variable</i>	<i>Empty was</i>	<i>Teacher led</i>	<i>Researcher led</i>	<i>Classroom</i>	<i>Adult Group</i>	<i>Additional comments</i>
<i>Bers, M.U., et al (2014)</i>	The use of developmentally appropriate robotics and programming tools by kindergarten children.	√	63	5	√	x	x	√	X	20	Children's concept mastery tended to be high but varied with concept difficulty 63 Enrolled data gathered for 53. By the 5th activity the percentage of students reaching target was 46% Teachers deliver lessons directed by CS
<i>Harms, K.J. et al,(2016)</i>	The reasoning behind learners selecting either tutorial or puzzles for programming.	∅	30	10	√	x	x	√	X	∅	Children cited their own perceptions of ease and difficulty using either tutorials or puzzles with a degree of struggle.
<i>Hill, C., et al (2015)</i>	Analysis of Scratch, ScratchJr, and Blockly block based programming languages	√	14 2	4th grade	√	∅	X	√	X	3	Identified a gap for 4-6 grade classrooms. for students and curriculum developers. Evidence is from two of the five schools shows variability and issues for children because of numeracy content. Not piloted in formal classroom.
<i>Hromkovič, J.,(2016)</i>	Describes approaches and experiences with teaching programming, shifting beyond the pure teaching of specifics towards sustainable topics.	∅	∅	∅	∅	∅	N o	√	∅	3	Initially lecturer led then children follow a booklet with teacher and assistant, No findings showing results.

Table 6 CS studies in primary extracted from the systematic review.

Table 7 and 8 show the number of participants, minimum age of participants, progress made, teacher or research led, adult to child ratio and classroom based in data driven studies √-Yes X-No ∅ - unable to determine from the publication * X – non-experimental studies √ quasi-experimental studies or ∅ insufficient information in the publication.

<i>Author(s) and Country</i>	Study outline	methodology *	Participants	Age (minimum)	Variable	control	Empty was	Teacher led	Researcher led	Classroom	Adult Group	Additional comments
<i>Ichinco, M. et al (2015)</i>	Exploratory study to identify the challenges of novice programmers using unfamiliar example code.	X	18	10	√	X	X	√	∅	∅		69% Participants correctly completed 37 tasks. Task completion rates varied between 1.63 minutes to 8 minutes.
<i>Lin, J.M.C., et al (2007).</i>	Investigating guided collaboration in elementary school students learning to program in MSWLogo.	√	64	∅	√	x	x	√	∅	2		SS difference in high achievers scores for collaborative approaches. No control group, but comparison made with guided collaboration, free collaboration, individual learning.
<i>Sapounidis et al 2013 (Greece)</i>	Children’s opinions across three age groups regarding a tangible and graphical isomorphic user interface.	∅	61	5	√	X	X	√	∅	∅		Measures first sight preferences of interfaces. Younger children found SS tangible interface easier to use. Older children preferred graphical.
<i>Sapounidis, T., et al 2015 (Greece)</i>	Exploring children age 6-12 performance on robot introductory programming activities with one tangible and one isomorphic graphical system	∅	109	∅	√	X	X	√	∅	∅		Volunteers randomly assigned.

The studies in this background reading section were available at the time of writing in 2016 and based on the original Szabo literature review search terms.

Table 7 Overview of CS studies in primary school contexts

Chapter 5

5.1 Introduction

This background reading in chapter 4 shows approaches in introductory programming tools and instructional method in primary education is scarce. The few data driven studies available are not set in authentic primary school classrooms. All classroom based studies are supported by high adult student ratio led by the CS researcher. This design has limitations for being upscaled or replicated in a typical classroom setting. In addition, given the significant challenge of training sufficient teachers in line with the implementation rates of CS in primary school curriculums there is a need to know how primary teachers without training implement the subject. This insight can support future work and resource development.

RQ 1: What typical programming approaches are in place in traditional primary classrooms and how successful are they?

5.1.1 Introductory programming

Introductory programming in this study is defined as the practice of teaching children, in a formal education setting, to program computers using a machine language for the purpose of problem solving.

This exploratory case study examines typical approaches to introductory programming in primary school. Research instruments and data for collection are informed by a focus group of teachers and an online survey. The field work involves 4 primary schools, 4 primary schools and 63 children. (Fig 1) Teachers are asked to plan a 6 week computer science course for their class. All choose to implement an introductory programming course. Children self-assess their levels of motivation for CS before during and after the course.

The study is organised as follows: The first section of this chapter in defines introductory programming for the purposes of this study.

It explains in section 5.3, the methods used for the exploratory focus group and online survey. Findings are organised into themes of children's learning experiences, curriculum

(CS concepts) and predictors of success. Conclusions from the online survey link to an exploratory case study involving 4 primary schools and 63 children.

The exploratory case study in the primary classroom methods.

This section provides an overview of participants, their age, the teaching methods adopted, and resources used during the 6 x 1 hour sessions; Lesson content in each school including a brief description of the programming environments; primary data scoring is explained and findings are organised into three themes; curriculum; motivation; and predictors of success (play interests).

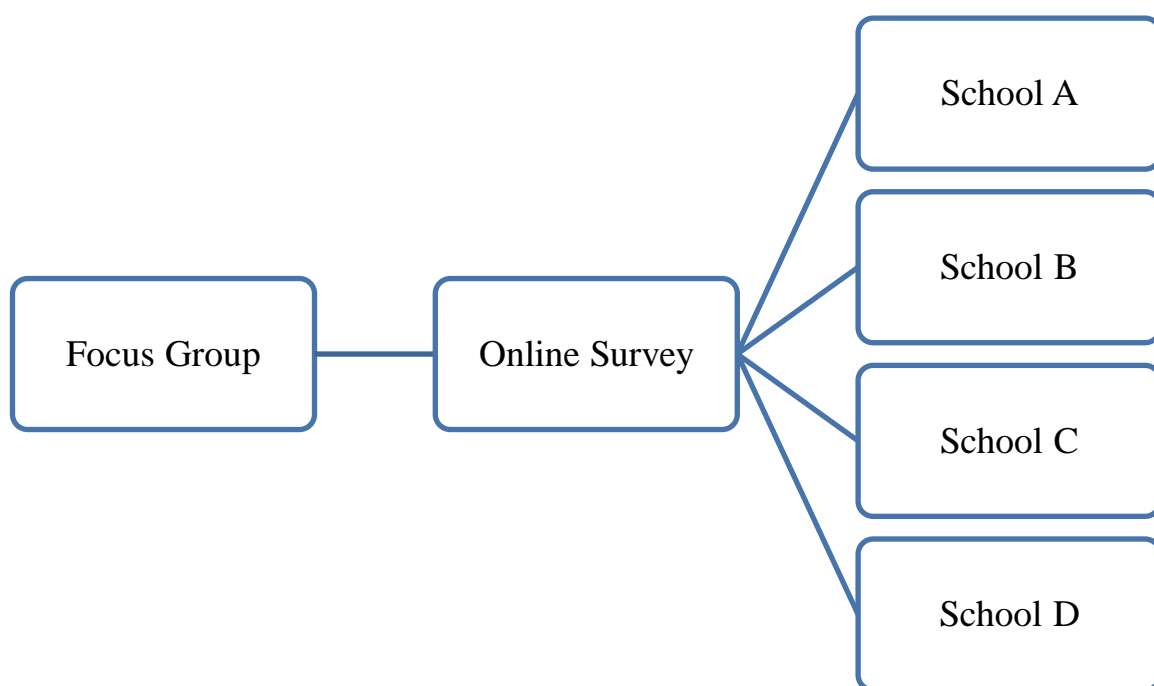


Figure 3 Shows the three-part study design for quadrant B

5.2 Method

Exploratory focus group and online survey

5.2.1 Instrument development - exploratory phase

The study involves preliminary work gathering information from practitioners experienced in delivering CS education in schools. Their views on typical methods children's success and predictors of success are explored in a focus group and inform an online survey. Data gathered from the survey informs the exploratory study in four primary schools. The survey was posted online through social media sites Twitter and Facebook. The survey was deliberately concise to improve completion rates. Its purpose was to focus on data gathering and provide insights for the classroom observation field study. The study gathers

information on the teacher's role, the student learning, CS concepts and predictors of success.

5.2.2 Focus Group

The study uses a focus group to develop questions for a survey distributed online for educators delivering introductory programming courses in school. The aim of the focus group session is to gain insight into RQ2 and inform the online survey instrument. Using snowball techniques, 6 participants experienced in delivering introductory programming courses in primary schools are interviewed individually for one-hour. The researcher presents 4 questions (Appendix B) linked to IP perspectives set out in Quadrant B background reading chapter of :

- teaching
- student and,
- curriculum:

The diagram below provides an overview of each of the themes covered within the study. Observing children’s learning in a classroom is complex and this framework provides an approach for organising the approach and themes.

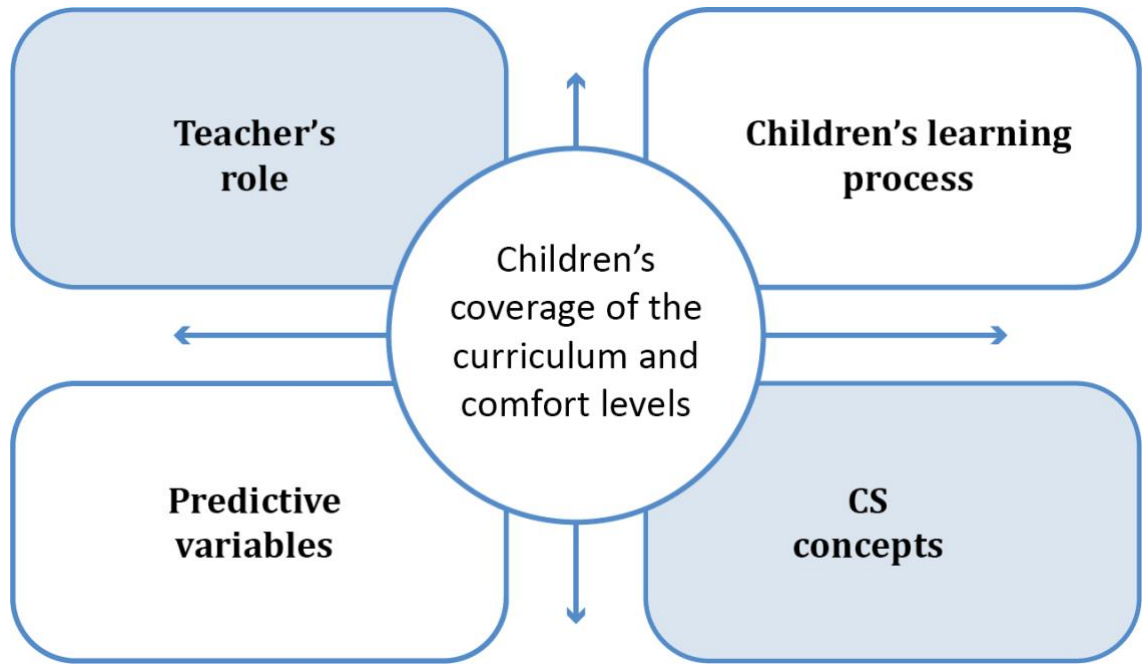


Figure 4 Shows contributing factors to learning in a primary classroom.

Phase one: Participants.

Six adults (educators) either leading or supporting learning in the classroom participated in the initial focus group interviews. The table below shows their role and background includes Primary teacher and visiting specialist such as a peripatetic instructor. The educator's background shows their qualification level and discipline. One instructor had no formal teaching qualifications or degree associated with computing science. However, they have significant experience working with young children teaching them computer science across the local authority.

PARTICIPANT ROLE	BACKGROUND	NUMBER	ADDITIONAL COMMENTS
Primary teacher	Art degree + PGDE	1	
Primary	B. Ed Primary	2	
Secondary teacher	CS degree + PGDE	1	
Peripatetic instructor	Work – no degree	1	The instructor works alongside primary teachers in school.
Parent volunteer	No qualification	1	

Table 8 Participants' backgrounds

Notes were taken from the meetings and amendments made to the questions for the online survey. The revisions were sent to participants who all unanimously agreed that it captured their discussions, and no important information was omitted. The focus groups evaluated the questions' accessibility, comprehensibility and all agreed that the survey questions answered the two research questions.

5.2.3 Data analysis – predictors of success

An open survey question asks teachers to report predictors of success for the highest and lowest achieving children. Using qualitative analytic method (Braun and Clarke 2006) five themes emerged; Attitudes; Academic; achievement in other curricula areas; outside interests; and personality. The themes were developed inductively. There was no predefined set of codes and a hierarchical frame was established. The process was validated through a proportionate sampling approach by peer PhD students. This verification highlighted two issues which were considered. The data was grouped to

ascertain a sense of what it looks like and the first codes assigned. Next the data was analysed line by line to code as much as possible. These codes were then placed into the coding frame. The results are shown in section 5.4.6.

5.3 Exploratory phase results

Demographics

67 participants indicate that they had experience of teaching introductory programming.

Table 9 shows respondent's sector background and the age range of children that they taught introductory programming.

- 20 of the 54 secondary specialist taught primary children. This is likely because of non-CS specialist primary teachers accessing secondary school expertise from their associated secondary school.
- All 22 primary specialists taught primary children and one of the primary specialists also taught children in the early years class. No primary specialist taught secondary introductory programming.

AGE	NUMBER	BACKGROUND OF RESPONDENT				% *
		ELC	PRIMARY	SECONDA RY	OTHER	
Of chn taught	Of respondents with experience teaching this age group					
3-5	2	1	1	0	0	50%
6-7	6	0	6	0	0	100%
8-9	12	0	7	5	0	50%
10-12	23	0	8	15	0	35%
12+	54	0	0	54	0	100%

*Percentage of children taught by a teacher with a specialist background or qualification for teaching that age group.
 ** Early learning and childcare setting

Table 9 Children's age taught and background of teacher

5.4 Phase one Results

5.4.1 Children's learning experiences

All 67 respondents with the exception of the early years specialist, use visual programming environments and 13% use unplugged activities. The early years specialist used programmable robots. 94% respondents used Scratch for introductory programming for children age 6 – 12+. Limitations with the survey design questions meant secondary teachers teaching both primary and secondary aged children (age 11-15) may not have answered specifically about the primary school children that they taught. This issue of the clarity in response from secondary teachers teaching across the two sectors resulted in challenges extracting primary specific materials. However, data shows that all primary specialists used Scratch. A few secondary specialist respondents used only Scratch for introductory programming when teaching both primary and secondary children.

CURRICULUM RESOURCES	RESPONSES	%RESPONSES
PRIMARY AND SECONDARY		
Scratch	63	94.03
Kodu	22	32.4
Hour of Code	18	26.87
Unplugged	9	13.43
Bee-bots	7	10.45
Alice	2	2.99

Table 10 Curriculum resources used for introductory programming

5.4.2 CS Concepts

28% of respondents commented on CS concepts they taught with variables, sequence, selection and loops being the most common. There was no clarity on progression frameworks supporting the delivery of planned learning. Two primary specialist respondents did not know the CS concepts that they taught. Most respondents did not complete this section.

CURRICULUM RESOURCES	RESPONSES	%RESPONSES
Variables	17	25.3
Sequence	17	25.3
Selection	17	25.3
loops	14	18.6
Other	12	11.8
Don't know	2	1.24%

Table 11 CS concepts taught reported by 28% of respondents

5.4.3 Teaching process

85% of teaching approaches include a demonstration at the front of the class with over half 53% directing learning through step by step. Qualitative data shows that most use a blend of approaches but start with the demonstration at the front of the class. This is typical of secondary specialists' approach to learning and teaching practice. (Table 12)

TEACHING APPROACH	RESPONSES	% RESPONSES
Demonstration in front of class	56	84.85
Direct support with a small group	17	25.76
Follow the leader step by step with teacher	35	53.03
Worksheet of instructions	42	63.64

Table 12 The range of teaching approaches

5.4.4 Success Rates

To provide insight for RQ2, the study explored respondents view of success during introductory programming. 16.42% of respondents reported that over 75% of the class achieve immediate success. Qualitative comments varied with one teacher commenting “some children just get it and others don’t” and “I can count on one hand the number of pupils who were able to achieve the ultimate aim independently with no support, though even these pupils required further support to tidy up their code. Only one respondent stated that over 75% of children needed high levels of support. They recorded in the comment box that this related to a group of children with significant additional support needs. Over 80% of respondents report that less than 25% of children require high levels of support and implies that from their view lesson implementation is appropriate for most children.

ASSESSMENT OF PROGRESS	RESPONSES	%RESPONSES
75 -100 % of children with Immediate success.	11	16.4%
75-100 % of children needing high levels of support	1	1.5%
Less than 25% of children need high levels of support	41	61.2%
Not answered	14	20.9%
Total	67	100%

Table 13 Children’s level of success

5.4.5 Attitudes towards learning

Children’s attitudes to learning impact on their progress across areas of the primary school curriculum. It is widely understood that motivation is a predictor of success (Salac and Franklin, 2020) and educator’s responses align with this view. All teachers report high levels of motivation by some students. 40% of respondents reported over 75% of children are highly motivated before and during the introductory course. 40% of respondents state that up to 25% of children show unexpected levels of motivation compared to other school subjects. 40% observed up to 25% of children continue to develop their skills after the course was finished. Educators responding in the survey comment that motivation within the lesson declined for most children and young people when they face a challenge, although this improved once they achieved success (Table 14)

Children’s levels of motivation	RESPONSES	%RESPONSES
75 -100 % of children show motivation	24	38%
1-25% of children show unexpected levels of motivation	26	40%
1-25% of children continue to develop their skills	26	42%

Table 14 Motivation levels during introductory programming courses.

5.4.6 Predictors of success

Most respondents commented on why some children were more successful than others. These comments were reported in isolation with themes linking to attitudes, success in other curricula areas such as maths and other interests outside formal education.

ATTITUDE PREDICTORS	
MOST SUCCESSFUL CHILDREN	LEAST SUCCESSFUL CHILDREN
Inquisitive	disengaged
Motivated	Low self esteem
Perseverance	Low aspirations
Growth mindset	Lack of curiosity
Creative	Unable to follow instructions
Take responsibility	Impatience
Concentration	
ACADEMIC/ACHIEVEMENT IN OTHER CURRICULA AREAS PREDICTORS	
MOST SUCCESSFUL CHILDREN	LEAST SUCCESSFUL CHILDREN
Problem solvers	Low literacy levels
Mathematical	High levels of support across all curricula areas
Logical Thinkers	Low numeracy
Chess	
OTHER INTERESTS OUTSIDE FORMAL EDUCATION	
MOST SUCCESSFUL CHILDREN	LEAST SUCCESSFUL CHILDREN
Computer games	
Puzzles	
Affinity with ICT	
Lego/construction	

Table 15 Views of predictors of success

PERSONALITY PREDICTORS OF SUCCESS	
MOST SUCCESSFUL CHILDREN	LEAST SUCCESSFUL CHILDREN
Autistic spectrum disorder	Gregarious
Isolated	
“Loner	
OTHER	OTHER
There isn't a type	Additional support needs
Boys	Girls
Innate	
When given opportunity to explore	

Table 16 Participants view of children's personality contributing to success in IP.

Conclusion

The survey shows that most educators delivering introductory programming courses to primary school children use Scratch visual programming environments and that progress varies. Various teaching methods include demonstration, step by step and worksheets. Participants views on predictors of success are attitudes, success in maths, interests outwith school and personality. One suggested gender and innate ability. This data provides an insight into curriculum, teaching approaches and student variables such as motivation. The initial survey shows a range of approaches to introductory programming in primary schools. Importantly, the sample shows few primary specialists deliver courses. Given the long standing issues across tertiary and secondary sectors, this study justifies triangulating the results in a real life primary school context. Perhaps primary teachers can ensure that all children succeed and are motivated?

All respondents made responses about children's interests outside formal education contributing to success in introductory programming. This view aligns with the Cutts Q., Patitas, E., et al study (2017) on children's early developmental experiences and

computing proficiency. These predictors of success could be explored further in the primary classroom setting to provide further insights into success.

5.5 Exploratory Case study in 4 primary schools

The focus group and survey results in phase one provide insights into a primary school approach to introductory programming. However, there remains a gap in data gathered during an authentic lesson. The survey results provide evidence that introductory programming is already in place in some schools. It helps shape the research instruments used in observing teachers and children in their authentic context. This includes gathering data about the type of resource used, the teacher's approach, the children's experience and the progress made. It also seems worthwhile exploring in the classroom setting, predictors of success.

5.5.1 Phase two - Methods

Field work observations of introductory programming in formal primary school classroom setting.

This phase two exploratory case study is planned and implemented a typical primary school learning environment. The research is conducted within a qualitative tradition (Miles and Huberman 1994) exploring the perspective of primary teachers implementation of introductory programming triangulated against observation and documentary data. The unit of analysis for the case study is the entire classroom learning environment including the role of the teacher, children's learning experiences and the course CS content. The study design takes account of the data gathered from the online survey mentioned above which provides insights into introductory programming in primary schools. To this end, qualitative data is gathered showing the resources used by the teacher for planning the course, children's motivation levels. Qualitative data for children's progress through the course and their interests in toys, games and activities outside school is quantifiable. Therefore, quantitative data is used for analysis of the children's progress and predictors of success from interests outside formal education.

5.5.2 The study

The study involves a 6 x 1 hour introductory programming course delivered in four schools to 63 children. Teachers plan, implement and assess the course. Children's motivation levels during and after the 6 week course alongside their interests in toys games and activities are recorded. Observing classroom practice in an authentic classroom is a complex phenomenon suited to this type of qualitative methodology

Participants for the fieldwork study volunteered through the online survey. Four schools (School A, School B, School C and School D) were selected from a rural local authority in Scotland whose central staff valued CS education as a means to support economic growth. The class teacher for all children involved in the study is present during the fieldwork. However, their roles varied depending on who was leading the lesson. In school A the class teacher plans and implements the learning. In school B and school C the learning is led by a visiting specialist with no primary teaching qualification but experience in teaching introductory programming across the local authority. School D was a primary teacher with college background. The 4 schools chosen are a random sample of the local authority although they include a mix of rural and urban settings. Table 17 shows the numbers of children in each class, their age and composition by gender (F = female M=male). The teacher and pupil ratio and the instructional design used whether demonstration at the front of the class and or use of worksheets. How the children were organised to complete the tasks in pairs or trios and the length of the course.

School	Class size	Age	Gender		Teach: Pupil	Method	Resource	group	Ind/	Duration
			F	M						
A	22	8-9	10	12	1:11	Demo	Wedo	Trio		6 x 1hr
B	21	8-9	9	12	1:21	Demo/ worksheet	Kodu	Pairs		6 x 1hr
C	9	7-11	6	3	1:9	Demo/ worksheet	kodu	Pairs		6 x 1hr
D	11	8-11	3	8	1:11	Demo/ worksheet	kodu	Pairs		6 x 1hr

Table 17 Demographics of schools involved in exploratory case study

Active participation consent is required for ethical approval including the purpose of the study to be shared with children using age-appropriate accessible language. All children and teachers agree to participate and consent forms are stored securely. Observations of teaching styles are made and recorded against a timeline. Children self-assess their motivation levels during and at the end of the course.

5.6 Measuring children’s progress – Quantifying the data

A maximum score of 30 is achievable across the six weeks. Each lesson has 5 data points linked to the teacher’s learning outcomes. These include computer science objectives and expectations for primary school children working collaboratively on tasks the data resulted in 4 data points contributing to the quantitative data for the analysis.

The study has four data points using data that was quantifiable.

- Data point 1: Children’s progress through the course
- Data point 2 – Children’s self-assessment of their learning
- Data point 3 – Children’s motivation
- Data point 4 – Children’s interests outwith school as predictors of success.

Children absent from any of the six lessons and children whose teachers had assessed their work inaccurately were removed from the study n=7.

5.6.1 Data point one - Children’s progress through the course

Each lesson planned by the educator has a purpose or set of expectations called the lesson learning intentions (LI) linked to individual success criteria (SC). The success criteria determined by the teacher includes computer science objectives and expectations for the children working collaboratively on tasks. One point is awarded for each of the SC children achieve. Each lesson has a set of 5 success criteria, some of which are repeated. A maximum score of 30 is achievable across the six weeks. The scores are split evenly across 6 bands from 1-30. The bands allowed data extraction of the highest achieving students for analysis using binary logistic regression. The detailed SC for each lesson is available in the appendices. School A: Appendix C and School B,C and D Appendix D

Assessment Banding

		Code
	Band 1= 1-6 points	B1
	Band 2= 7-12 points	B2
Progress score	Band 3 13-18 points	B3
Band 5=1	Band 4= 19-24 points	B4
Less than Band 5=0	Band 5= 25-30 points	B5

Table 18 Success criteria assessment banding for total scores

5.6.2 Data point 2 – Teacher’s predictions of children’s progress

The table below shows the statements used by teachers to determine if children achieved or exceeded what the teacher expected them to achieve or exceeded during the introductory programming course.

Statements teachers used to assess children’s progress.	SCORE
DESCRIPTOR	
Child made expected progress	0
Child achieved better than expected progress	1

Table 19 Scoring for children expected and actual progress

5.6.3 Data point 3 – Children’s motivation

Children’s self-assessment of their motivation levels on a scale of 1 to 5 towards computing science prior to the course, during the course and on completion of the course.

Table 20-21 shows the statements used to describe children’s motivation before the course, during the learning and at the end of the introductory programming course was taken. Prior to the course starting, children scored their motivation levels using a scale from 1 – 5. A score of 1 is recorded by the children if they do not enjoy playing computer games. They can score 5 if they play computer games constantly with high levels of enjoyment. A specific focus on computers was included to distinguish enjoyment levels of lego and puzzles and digital technologies as a separate entity.

DESCRIPTOR	SCORE
MOTIVATION PRIOR TO THE COURSE	
I don’t like playing computer games at all.	1
I like playing a bit but not too often	2
I like playing computer games but prefer other games and toys	3
I play computer games a lot of the time.	4
I enjoy playing computer games and play all the time. I make my own games.	5

Table 20 Motivation level scoring prior to starting the introductory course

DESCRIPTOR	SCORE
MOTIVATION DURING THE SIX LESSONS	
I didn’t enjoy learning to program.	1
I enjoyed learning to program a bit but each week there were times that I didn’t enjoy what I was doing.	2
I enjoyed learning to program every week and sometimes I found I didn’t enjoy it.	3
I enjoyed learning to program every week.	4
I enjoyed learning to program every week and looked forward to the next week.	5

Table 21 Motivation level scoring during the introductory course

DESCRIPTOR	SCORE
MOTIVATION AFTER THE SIX LESSONS	
I would not like to do more programming in school or at home.	0
I would like to do more programming in school or at home.	1

Table 22 Motivation level scoring after the introductory course

5.6.4 Data point 4 – Children’s interests outside school as predictors of success

The multi-dimensions involved in observing children learn in their primary classroom in this study are considered through the teacher/educator’s role, the children’s learning experiences, the CS curriculum and the predictors of success (Fig 4).

Bergin and Reilly (2006) claim identifying predictors of success in the first few weeks of HEI students IP course could help considerably to alleviate challenges. Therefore, the thesis includes a focus on predictors of children’s success in CS IP. In other subjects, such as literacy and numeracy, there is a relationship between children’s early experiences contributing to their later interests in a subject (Bush, L., 2003). Bush describes the relation between children’s experiences with books and print influencing their ability to comprehend what they read. In addition, children’s learning through listening and talking contributes to their ability to read and write and vice versa (Strkland, D.S. & Riley-Ayres, S., 2006). Cutts, Q., Patitsas, E., et al (2017) also propose a link to childhood experiences and aspects of computing proficiency. Therefore, it is of interest to the thesis focus on success in IP to explore this contributory factor. The term teacher and or educator is interchangeable and refers to the adult leading the learning.

Children wrote about the types of toys, games and activities that they enjoyed outside school. These interests were categorised qualitatively using Whitebread (2012) play categories framework of physical play (PHY), playing with objects (OBJ), symbolic play (SYM), pretence play (PRE), Games with rules(GAMES) and Digital technologies (DT).

CODING: PLAY INTERESTS BASED ON WHITBREAD'S PLAY CATEGORIES.	CAT
Physical: Active exercise: Rough and tumble: Fine motor practice	PHY
Objects: Building, making and construction	OBJ
Symbolic: Reading, writing, number, painting, drawing, collage, music.	SYM
Pretence: High-quality pretend play: Dolls: Roll play.	PRE
Games with rules: Including making own games	GAMES
ADDITIONAL CATEGORY	
Digital technologies	DT

Table 23 Categorisation of play interests for data point 4

The pre and post intervention surveys differ because of the context. The children had no experience of programming therefore the pre-questions focused on the technology. The post questionnaire was of interest because of the children's motivation to continue. These scores were not used statistically or to correlate. The survey design was to establish if children by the end of the intervention are interested in programming after their first exposure to the CS experience.

5.7 Results

5.7.1 Teacher's role

Researcher's observation notes and teacher's plans were reviewed and show the following information:

School A: The teacher planned a series of lessons for the 6 x 1 hour introductory programming course. The lesson learning objectives are informed by Lego WeDo robotics instruction cards created by the manufacturer. The plans did not reference computing science concepts. Teachers introduce the work from the front of the class and circulate the classroom answering questions as they arise.

Schools B, C and D: One participant downloaded planning resources from the local authority website (Argyll and Bute no date) which were shared across all three schools. The plans link to an online visual resource Kodu. Teachers introduce the work from the front of the class and circulate the classroom answering questions as they arise.

5.7.2 Children's learning experiences

Researcher's observation notes were reviewed and show the following information:

School A: Lego WeDo

Teachers created a series of lessons based on Lego WeDo robotics. They used instruction cards from the manufacturer to plan lessons and core learning objectives. No computing science concepts were identified by the teachers. In week one, children follow a series of steps set out on the instruction cards to build and control a robot. Once the robot is complete the user creates a program for the robot to solve a problem. Children work in pairs, they copy the basic programming language set out on the instruction cards to controls the robot.

School B, C and D : Kodu

Teachers implement the same Kodu lesson plans downloaded from the national improvement agency Education Scotland. Children follow the step by step instructions created by the class teacher. They create a computer game within a 3-D world. Children select tiles and connect them to execute the program. Each tile is a component part of the program.

In this study, educators plan the introductory programming learning using Lego WeDo (Mayerová 2012) and Kodu (Stolee and Fristoe 2011) visual programming environments. Below is a brief overview of two resources.

Lego WeDo

Programming Environments

Lego WeDo: Is used in school A. The set, designed by Lego includes small plastic pieces and mechanical parts including robot bricks, sensors, LEGO USB hub and a motor. The set comes with easy-to-use icon-based drag and drop software interface providing an intuitive programming environment with building instructions, programming example.



Figure 5 Screenshot showing Lego WeDo on the screen drag and drop interface



Figure 6 Image showing Lego WeDo robot.

This robot has the control capability to move the helicopter propellers through cylindrical and belt drives.

Kodu

Kodu is a visual programming language used by schools B-D that is entirely event driven. The programmer creates a computer game and creates a 3-D virtual world through the understanding of code whose elements are image icons (Iskrenovic-Momcilovic 2018). It involves placing programming tiles in meaningful order to form conditions or actions for each rule. Despite its simplistic programming model many CS concepts can be expressed in Kodu.

5.7.3 CS curricula

School A: Lego WeDo

The curricula involves children copying solutions directly from the construction cards to build a solution. For example, task one requires the children to create a robot that stops a goal being scored. Learning relates directly or partially to the robotic kit and teachers do not overtly link or reference CS concepts.

School B, C, D: Kodu

Children are introduced to features and functions of the Kodu application and in week one begin building a virtual gaming environment. Learning links to the CfE experiences and outcomes, the planning formats state that children will convey information describe events, explain processes or combine ideas. This learning outcome from the national curriculum

was not observed during the 6 week session. In addition, written evidence in teacher's plans and lesson observations show that there is no explicit coverage of CS concepts.

5.7.4 Progress measures and scores

This section reports upon the four quantitative data points.

- Data point 1: Children's progress through the course
- Data point 2 – Teachers expected progress of children
- Data point 3 – Children's motivation
- Data point 4 – Children's interests outside school as predictors of success.

5.7.5 Progress scoring

A maximum score of 30 is achievable across the six weeks. Each lesson has 4 data points linked to the teacher’s learning outcomes (see section 5.7).

Out of 63 sets of complete data, 26 (38%) of children score within the range 25-30 (band 5) in each of the school. The majority of the 26 children achieving band 5 are boys n=20, 6 are girls.

SCHOOL	MEAN	STN DEV	MIN	MAX	COUNT
A,B,C,D	22.8	6.65	11.00	30.00	63.00
A	24.0	5.86	12.00	30.00	22.00
B,C,D	22.24	7.02	11.00	30.00	41
B	24.2	6.50	12.00	30.00	22.00
C	22.22	6.86	11.00	30.00	9.00
D	19.63	7.6	11.00	29.00	11.00

Table 24 Results for each school

SCHOOL	Band 5 (high)-1 (low)				
	5	4	3	2	1
A	9	9	4	0	0
B	11	4	3	3	0
C	2	3	2	2	0
D	4	2	4	1	0
Total	26	18	6	13	0
TEACHER ASSESS					

Table 25 Number of children achieving each band based on weekly SC scores

A similar proportion of children achieve band 5 across each school. Therefore, in this study using tangible objects (School A) did not improve scores. However, there are higher rates of motivation to continue on completion of the course in school A which used tangible robots in comparison to schools B. A few children in each school did less well than expected by the teacher. These children are successful in most areas of the primary school curriculum and this surprised the class teachers. In addition, at least one child in each school achieved more than was expected by the teacher based on their academic achievement in other areas of their learning.

5.7.6 Motivation levels (Data point 3)

Children self-assessed their motivation levels prior, during and on completion of the introductory programming course. 54 (85.7%) of children, irrespective of their scores, are motivated by the course and want to continue. The results table below shows motivation levels generally decline prior to starting and during the course. However, the categorisation is not comparable and should be analysed separately. It is clear from the data that all children are motivated by the expectation of programming, however, across the four schools by the end of the course 10 children do not want to engage in further computing science learning.

School	Motivation prior					During					After	
	5	4	3	2	1	5	4	3	2	1	C*	% of n
A	9	9	0	5	0	3	2	10	7	0	18/22	81
B	11	4	0	1	3	7	0	5	9	3	16/21	76
C	2	3	0	1	0	0	1	5	3	0	9/9	100
D	4	2	0	0	2		2	1	6	2	10/11	81
Total	26	18	0	8	5	15	5	21	25	5	53/63	ave = 84.5

Table 26 Children's motivation levels.

Table 26 shows Numbers of children and their self-assessed motivation levels before, during and after the 6 week course. * Number of children per cohort (n) indicating that they want to learn more computer science.

The data is analysed further taking account of gender. The study comprises of 34 male children and 21 females. Of the 26 children achieving band 5, 22 want to continue, the 4 children in school A who do not want to continue are girls. Interestingly, 12 out of the 13 achieving band 2 (score 7-12) want to continue, they include male and female. The one child achieving band 2 who did not want to continue was female.

SCH	Pupils		Score		Cont	Gend cont	
	Tot num of pupils by gender		5		Yes		
	M	F	M	F		M	F
A	12	10	6	3	8	6	2
B	14	7	9	2	9	9	0
C	4	5	2	0	2	2	n/a
D	4	7	4	0	4	4	n/a
Total	34	29	21	5	21	21	2

Table 27 Gender and motivation level of band 5 children

5.7.7 Play interests Data point 4

The focus group and online survey with educators explored participants' views on children's attributes or interests that predicted success in introductory programming. This area of interest is explored through a thematic analysis of children's responses to their reported play interests. The deductive themes framework for the analysis used Whitebread et al (2012) play categorisation. Due to the inductive nature of thematic analysis, an additional category of Digital Technologies was added. Each category was a variable for the binary regression with the dependent variable children who achieved Band 5 in the progress through the course score.

The Goodness-of-Fit Tests for the binary regression show The Nagelkerke R square is .260 Hosme and Lemeshow Test P is .612 . Findings show a significance with physical play and the Exp(B) odds ratio of 7.862 which is greater than 1 suggesting increasing odds of the event occurring. The regression also returns a close to significance level with games with rules and an odds ratio of 3.527 suggesting that increasing odds of the event occurring. Data gathered linked to children's motivation levels, gender, play interest and progress scores is shown in Table 28. However, the limitations of this element of the study are acknowledged.

		Variables in the Equation					
		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1 ^a	Physical	2.075	.905	5.251	1	.022	7.962
	Objects	.643	.669	.922	1	.337	1.902
	Symbolic	.294	1.006	.086	1	.770	1.342
	Pretence	1.118	1.534	.532	1	.466	3.060
	Games with rules	1.261	.701	3.232	1	.072	3.527
	Digital games/toys	.675	.633	1.139	1	.286	1.965
	Constant	-3.019	1.002	9.075	1	.003	.049

a. Variable(s) entered on step 1: Physical, Objects, Symbolic, Pretence, Games with rules, Digital games/toys.

Table 28 Regression output for highest achieving children and play interest.

5.8 Discussion/Future work

Teacher's views gathered from the online survey successfully predict outcomes arising from the fieldwork. Findings across all four schools show that teachers plan children's learning based on a create-first approach. Children follow step by step worksheet approaches with the resources driving the learning and teachers expressing limited knowledge on CS concepts being covered. Progress through the course shows, children's success rates vary. Males score higher than females and more boys are motivated after the course to continue to learn programming. A few children with low progress scores self-assess as highly motivated. A few high scoring females do not want to continue programming after the course is completed. The study explored predictors of success using children's other interests. However, no statistically significant relationships with children's reported play interests and their progress across the 6 week introductory programming course are found. The study highlights issues in the lack of CS concepts used in teacher's assessment. Future work suggests an exploration of introductory programming in primary schools using research informed practice. Limitations from the study design suggests improved data gathering research instruments are needed. All four schools and participants in the online survey use block programming environments with primary age children in formal education. This aligns with literature that reports block-based programming languages are easier for children than text based programming languages within the CS curricula. The study shows introductory programming course delivered by primary specialists in school A and D with an instructor delivering the lessons in schools B and C. Both the primary specialists sought advice from teachers in the local secondary school on the materials that they planned. The backgrounds of the educators delivering the course did not impact on children's progress. All children were expected to build solutions immediately.

The data gathered in the study reflects a normative curve showing children progress at different rates. In addition, there appears to be gender differences with a link between gender and success rates. More males achieve the higher scores than females. Most children enjoy the course and report high levels of motivation, although motivation levels after the course are higher in males. However, there is an issue for a few children who did not want to continue and interestingly, a few children achieving less well are motivated by the approach. In addition, the teacher's predictions are that a few children achieve less well than expected, although a few children achieve more. Males showing higher levels of

motivation. The results from the data collection provides conclusive evidence that children's success rates vary during introductory programming courses.

Findings from the exploration of children's interests outside formal schooling and progress, show links with physical play and games with rules. It is acknowledged that the data gathering techniques for these findings need improved. However, given the rule bound nature of programming then this is a probability. At this stage the physical play result is non-conclusive.

Finally, the initial focus group and online survey provide an additional layer of evidence from a wider population supporting the findings in the case study.

Future Work

The curriculum, resources and teaching approaches varied across the four schools although all educators did adopt a create-first approach with children building solutions without understanding the programs. This is a key point in the thesis and can be exemplified in school A where children were given a worksheet to follow and build the WeDo robots with no exploration of their understanding or the scripts they created and how these controlled the robots. In schools B, C and D all children followed the instructions to create the script. This step by step approach did result in a functioning game but children did not have the opportunity to share their understanding of the blocks functions.

The observation notes show variations in children's progress, motivation and perceptions. There is evidence to consider that some children underachieve and are demotivated by the experience. It is therefore recommended that another field study is undertaken with a more informed approach to introductory programming. In doing so there is a need to have greater control of the curriculum content, resources used and teaching approaches. Given the use of a create-first approach adopted by educators in this study, there is justification to use an alternative approach such as comprehension-first approach. With the success of comprehension-first in HEI, a worthwhile next step is exploring the approach in a primary context. In addition, a move from teacher's holistic judgement of children's progress to more rigorous measures.

RQ 3: Can a comprehension-first oriented introductory programming course be implemented successfully in a primary school classroom?

5.9 Limitations

While significance was shown with physical play (PHY) and games with rules play interests, the gathering of this data lacked rigour. The study did not control conditions such as the curriculum, resources and teaching approaches. Teacher's assessment of the introductory course provides a broad view of progress. Assessment was informed by online commercially produced packages and did not link to the national curriculum or focus on CS concepts. This potentially impacted on non-cs specialists' delivery of a CS curricula.

Quadrant C: Research informed practice in primary school introductory programming.

Quadrant B explored outcomes for introductory programming in a traditional classroom setting. Findings from the literature and study suggest that more insights are needed to support the future development of professional learning materials. Quadrant C explores a research informed approach to introductory programming and is organised in two chapters. At the time of writing, the Scottish computing science curriculum for all children age 3-15 is published (Education Scotland, 2017) . Chapter 6, explores the new Scottish curriculum recognising the importance of comprehension as the underlying approach. The chapter concludes that there is a gap in literature on the effectiveness of a comprehension-first approach in primary schools with few CS experts with primary school- expertise and few primary school experts with CS expertise. This leads to a study in chapter 7 where Ambassadors from university with no primary school knowledge develop comprehension-first materials moderated by a CS professor and primary school expert. These materials are implemented by the 2 Ambassadors in 2 different schools with 3 cross age classes. A total of 6 classes of children participated each lesson was observed by the class teacher.

RQ 3: Can a comprehension-first oriented programming course be implemented successfully in a primary school classroom?

Chapter 6

6.1 Quadrant C

Quadrant B introductory programming study explored outcomes for children in a traditional classroom setting. Findings from the study suggest that more insights and collaborative approaches with primary teachers and the CSED research community are needed to support the future development of professional learning materials. However, at the time of writing, in Scotland, Curriculum for Excellence (CfE) CS standards were issued, and it was therefore unlikely to find teachers in primary school with full CS knowledge or CfE CS knowledge experts with a deep understanding of primary school education. The revised CS curriculum moves from a content creation-oriented curriculum to a concept driven comprehension-oriented approach. In essence, the CfE CS curriculum promotes comprehension of code before writing code. A few instructional designs in place based on a comprehension-oriented approach have been successfully trialled in secondary schools. Therefore, a pilot study was planned using adapted materials, created by CS Students learning about CS Education, in a traditional primary classroom. 2 CS students planned and implemented materials to 3 cross age classrooms in 2 different schools. Each lesson was observed by the classroom teachers. The study involved a total of 2 Ambassadors, 6 primary school teachers and 158 children.

6.2 Related Work

6.2.1 The Scottish Curriculum for Excellence

A ‘National Debate on Schools for the 21st Century’ and a review of the whole educational system in Scotland (SEED,2002; Munn et al, 2004) aimed to identify what was the value and purpose of education, and what was hindering progress and social equity. The aim was to improve the national educational experience, achievement and attainment for 3–18-year-olds and beyond, considering skills for learning, skills for work and skills for life (Scottish Government, 2009b) (McLaren, 2011).

Scotland’s Curriculum for Excellence (CfE) was formally implemented in 2010-2011 emphasising generic skills and competencies, a focus on pedagogy and an ambition that young people are expected to develop as successful learners, confident individuals responsible citizens and effective contributors as a result of their school education (Scottish Executive, 2004). The Curriculum is grouped in linear level through loosely framed Experiences and Outcomes, that seek to specify not only the result, the outcome of

learning, but also the experience undergone by the pupil in attaining the outcome (Priestley and Minty, 2013).

The curriculum reflects what Scotland values as a nation and what it seeks for young people. It is designed to convey knowledge considered to be important and to promote the development of values, understanding and capabilities. It is concerned both with what is to be learned and how it is taught. The curriculum should enable individuals, reach high levels of achievement, and make valuable contributions to society (Scottish Executive, 2004).

The overarching structure of CfE is described in terms of eight curricular areas - expressive arts, health and wellbeing, languages, mathematics, religious and moral education, sciences, social studies and technologies. The traditional subjects have clearly defined bodies of knowledge with well-established methods of investigation and standards for determining the validity of new knowledge. The technologies subject area acknowledge specialist disciplines of Business, Computer Science, Home Economics, Technical/DT, there are business, computer science, textiles and food contexts, and craft, design, engineering and graphics contexts for developing technological knowledge and skills. It is this computer science specialist area that this study refers to within the broader area of the technologies.

In 2017 the Curriculum for Excellence technologies subject area was updated to ensure learners between 3-18 have the knowledge, understanding and skills in the technologies required to succeed in today's world (Education Scotland, 2017). The revised framework provides educators with the underlying principles and practice for teaching computing science. It includes CS experiences, and outcomes to plan learning and CS benchmarks for assessing children's progress. The refreshed CS curricula coincided with the next phase of the thesis and required consideration when planning materials for implementation in the classroom.

6.2.2 The Curriculum for Excellence computing science approach

In 2017, the CfE CS curricula in Scotland radically changed moving from a typical content led approach to what could be described as a concept led future proofed approach. Prof. Quintin Cutts, University of Glasgow, Prof. Richard Connor, University of Strathclyde and the Prof. Judy Robertson University of Edinburgh worked with teachers to develop ideas and support the changes (Farrell et al. 2017). They consider that the revised curriculum focuses not only on computational thinking but includes knowledge of computers. It is important to teach children, at a developmentally appropriate level, about CT in everyday

life, machine architecture, the semantics of programming languages and the building new software. The CfE framework organisers consist of experiences linked to:

- Domains in which computers operate within real life contexts.
- Computational mechanisms that make computers work, for example, visual programming environments such as Scratch.
- How to use computational mechanisms, children create scripts or simple programs.

Other countries school level curriculum and resources such as ‘Scratch’ arguably focus mainly on the third aspect in the form of writing programs. However, the exploration of common computational processes with real world applications or understanding how computers execute instructions in programming languages takes less importance in primary school. For CS to take its intellectual place all three foundational concepts are considered important to be taught at an early age. The CfE CS approach supports curriculum organisers by including the three key aspects of CS education identified by the researchers within a broader framework of the following key concepts:

The CfE concept “Understanding the world through computational thinking” (TCH 0-13a to TCH 4-13b) corresponds to what the research refers to as

‘(1) domains that can be modelled by computational mechanisms’.

The CfE concept ‘Understanding and analysing computing technology’ (TCH 0-14a to TCH 4-14c) corresponds to what the research refers to as

‘(2) computational mechanisms themselves’.

The CfE concept ‘Designing building and testing computing solutions’ (TCH 0-15a to TCH 4-15a) maps corresponds to what the research refers to as

‘(3) how to use the computational mechanisms to model aspects of the domains’.

6.2.3 CfE introductory programming in the primary classroom

The revised CfE approach brought challenges. Importantly it is unlikely that primary school teachers have a deep knowledge of CS and those with a deep knowledge of CS are unlikely to have a deep knowledge of primary education. However, as mentioned previously the Scottish CSED research community provided professional learning and published support material in the form of a handbook (Farrell et al, 2017) and professional learning sessions.

The Teach CS Primary Guide manual (Farrel et al., 2017) explains the three concepts of the revised curriculum alongside example activities for primary teachers to implement the subject and embed the changes. It emphasises the spiral nature of the CS curriculum and how it aims to meet curriculum aspirations more broadly by engaging learners in problems, theories, practices that are representative of those in the real world and computing education specifically. The guide does not, however, provide instructional models specifically for teachers who may plan for the creation-oriented approach as observed in quadrant B.

6.2.4 The create-first approach

The create-first approach can be defined as an approach where children create visual programming scripts through exploration of the blocks. Visual programming environments lend themselves to this approach and the create -first is evident in primary school age studies. In a study with young children using robotics authors investigated basic programming concepts to a summer school of children age 9-12 very quickly create codes to instruct Beebots (Athanasiou, Topali and Mikropoulos, 2016). In a study (Sáez-López, Román-González and Vázquez-Cano, 2016) authors evaluated the use of a Visual Programming Language using Scratch in classroom practice. The outcomes and attitudes of 107 primary school students from 5th to 6th grade in five different schools in Spain were analysed. The instructional design is not clear, however, it appears that children create scripts very quickly using an exploratory approach. The create-first studies above appear to reflect the perhaps common approach to introductory programming. In these examples, children are taught through code writing activities with the main goal being the development of a program. However, in some studies the approach is less clear, Funke, et al., (2017) evaluate games made by children using Scratch. Where unplugged activities act as the precursor, it is not clear if a comprehension-oriented approach is used.

A few issues can arise with the creation-oriented approach such as, students may get stuck for long periods of time. Where students copy examples of code it can quickly go from eyes to fingers bypassing the brain. When students spend long periods of time typing code, the number of examples used in a lesson to reinforce learning can be small with large amounts of time typing.

6 The comprehension-first approach

The alternative approach from create-first is a comprehension-first with code reading taught explicitly before writing. As mentioned previously, the Scottish curriculum framework supports the theory that there is a greater need for emphasis on comprehension

before writing code. It moves away from the pattern of creation orientated approach evidenced in some primary school literature. In a comprehension-oriented approach, however, it is difficult for students to become completely stuck; examples are studied to successfully complete the activity and the number of examples used to reinforce learning can be much larger (Waite, 2017). There are a few comprehension-oriented instructional models such as PRIMM (Sentance and Waite, 2017), Use Modify Create (Lee et al., 2011) Parson's problems (Morrison et al., 2016) and specifically within the Scottish context Plan C professional learning models (Cutts et al., 2017) However, at the time of writing they are used most often in secondary school contexts with text-based programming languages. Research of comprehension-oriented approaches in primary school contexts is limited and it is worth exploring further.

Predict, Run, Investigate, Modify and Make (PRIMM)

PRIMM (Sentance, and Waite, 2017) is a suggested approach for working with beginners using text-based programming and more recently there is evidence of the approach for primary and block based languages. The framework is code comprehension based with the need to read code before it is written. In addition, authors propose that tracing code accuracy before introductory programming students can write code independently with confidence. The approach is based on Schulte's (2008) block model which is an educational model of program comprehension as a tool for a scholarly approach to teaching. (Sentance and Waite, 2017).

Using a PRIMM approach, teachers ask children to predict the outcome or *function* of a working program before running the program to test out predictions. Next children investigate the *structure* or meaning of each line of code through tracing or annotating, explaining or talking about individual parts before modifying and editing the program to do different things (function) then making their own. An essential element of PRIMM is not copying code, it is using starter code provided by the teacher and predicting what it might do before running it. At the time of writing PRIMM was mainly used in secondary schools although more recently is evident in primary school contexts.

Use modify Create

The Use-Modify-Create progression (UMC) (Lee et al., 2011) approach is, "based on elements from Experiential Learning Theory wherein knowledge is created through the transformation of experience, and Social Constructivism, which posits that through discussion and collaboration, students construct their own knowledge from experiences that have personal meaning to them". UMC, describes a pedagogical framework for

teaching computing in the domains of robotics and game development. In the “Use” stage, learners are consumers of someone else’s creation. Over time, in the “Modify” phase, learners alter the computational artefacts with increasing levels of sophistication. As learners gain skills and confidence, they are encouraged to develop ideas for new computational artefacts of their own design that address issues of their own choosing. Within this “Create” stage, abstraction, automation, and analysis come into play. UMC is used in secondary and tertiary education. It also uses the principle of starting with code that does not belong to the learner thus reducing the burden on students to create something that works straightaway.

Parson’s problems

Learning to program using text-based language is complex and can be frustrating for students. The principle behind Parson’s puzzles (Parsons and Haden, 2006) is to provide students with correct lines of code that are jumbled. Students select and rearrange code fragments by placing them in the correct place in the solution to solve the puzzle. This approach enables learners to practise writing code without the additional challenge of remembering program syntax. When used well, students enjoy the approach, it provides immediate feedback and can improve learners’ understanding of programming constructs.

Plan C Professional Learning Model

Plan C was a professional learning model supporting networks or ‘Hubs’ of secondary school computer science teachers in Scotland. Research informed practice covered in lead teacher training and local hub sessions focused on a range of topics including the Block Model of code comprehension (Schulte, 2008). Programming was primary context for delivering the topic, similar to the approach of the CfE CS 3-concepts, it is relevant to the learning of any computer system involving a language of instruction and an underlying computing engine, such as database and web systems (Cutts et al, 2017).

Conclusion

The Curriculum for Excellence computing science curricula provides a framework that lends itself to a comprehension-oriented approach. Block based programming languages are currently advocated as being the most appropriate type of programming environment for young learners with a prediction that this will remain so for the foreseeable future (Kolling, 2015). Findings from Quadrant B show primary teachers in the exploratory

study adopted a creation-oriented approach planning learning that expected children to create scripts and potentially not understanding the underlying concepts.

A few instructional designs such as PRIMM, UMC and Parson's problems involve students using pre-written scripts as a basis to reinforce learning (Waite, 2017) and align more to the CfE CS framework. However, at the time of writing, few of the materials had been trialled in a primary school context. This point is supported by Lye and Co in 2014 in a literature review of the K-12 field. They concluded that further research should be situated in class settings as less than half of the studies synthesised in their review were in primary or secondary classrooms. With this gap in mind, a pilot study was designed to implement modified materials in the primary school classroom.

Chapter 7

The research informed practice study

7.1 Introduction

Understanding the impact of introductory programming on primary school age children is at an early stage by comparison to subjects such as literacy, numeracy and STEM. This takes account of the literature gap identified in chapter 5 in insights on primary school pedagogy. Studies involving the create-first and studies involving the comprehension-first approaches are emerging. However, papers comparing both methods are scarce. The Scottish CS CfE curricula introduced in 2017 aligns with instructional designs such as PRIMM, UMC and Parson's problems. These comprehension-first approaches involve students using pre-written scripts as a basis to reinforce learning (Waite, 2017). However, at the time of writing, few of the comprehension-first materials had been trialled in a primary school context.

7.2 Method

7.2.1 Aims and Objectives

Quadrant B introductory programming study explored outcomes for children in a traditional classroom setting. Findings from the study suggest that more insights and collaborative approaches with primary teachers and the CSED are community are needed to support the future development of professional learning materials. However, at the time of writing, in Scotland, Curriculum for Excellence (CfE) CS standards were issued, and it was therefore unlikely to find teachers in primary school with full CS knowledge or CfE CS knowledge experts with a deep understanding of primary school education. The revised CS curriculum moves from a content creation-oriented curriculum to a concept driven comprehension-oriented approach. In essence, the CfE CS curriculum promotes comprehension of code before writing code. A few instructional designs in place based on a comprehension-oriented approach have been successfully trialled in secondary schools.

RQ 3: Can a comprehension-first oriented introductory programming course be implemented successfully in a primary school classroom?

7.2.2 Approach

The Curriculum for Excellence (CfE) Computing Science (CS) curricula provides an exploratory case study is set in a formal primary school context.

Qualitative and quantitative data gathered from the study provides evidence for the findings. Data sources is as follows with an overview in Figure 3. Convergent parallel design methodology is used to collect qualitative and quantitative data simultaneously and independently. In the analysis stage equal weighting is given to the quantitative and qualitative data. The results are then compared and contrasted looking for patterns or contradictions.

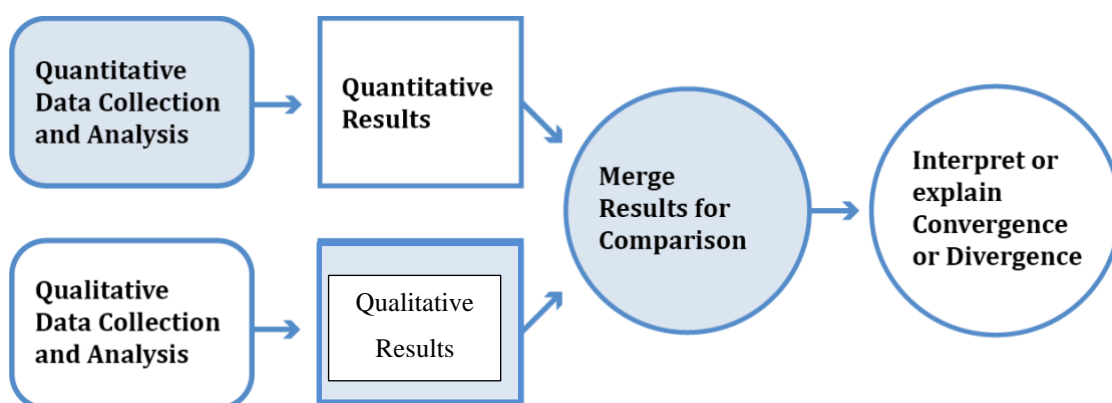


Figure 7 Research design focus to inform focus

7.2.3 Research Design

Understanding the impact of introductory programming on primary school age children is at an early stage by comparison to subjects such as literacy, numeracy and STEM.

Therefore, this exploratory research design is suitable because at the time of writing the research question has not previously been studied in depth.

Children’s CS learning is complex and observational techniques are best suited to explore teacher’s approaches and outcomes on learning in a natural setting. Observational techniques is to describe a variable or set of variables. More generally, they are used when the goal is to obtain a snapshot of specific characteristics (Brophy, 2006).

7.2.4 Setting

The study takes place in children's primary school classrooms.

7.2.5 Participants

The ambassadors

University of Glasgow final year CS undergraduate students (Ambassadors) undertaking the 'computer in the classroom' module plan and deliver the learning. As part of their assessment the students undertake a placement in a Scottish school. The students choose a primary school placement. The ambassadors had no primary school teaching experience.

The schools

A teacher known to the University of Glasgow with connections in two primary schools secured each headteacher's agreement in principle to the study. The initial contacts were followed up with the Director of Education of the local authority council for approval and to confirm participation. Each headteacher identified the three different age group classes involved in the study. Parental consent was sought. The children's class teachers have no computing science experience; they observe the lessons and support children's self-assessment. This introductory programming course was each child's first experience of introductory programming in formal education.

The study takes place in two self-nominating city primary schools in an area of high socioeconomic status with low levels of deprivation. Although the schools were selected through snowballing techniques, it is worth noting the high SES. Each school nominates three classes across three stages of the school. 158 children age 7-10 are involved in the study.

The table below shows the composition of the children in school B and school M. School B Ambassador is male and school M ambassador is female. Participants in the table are organised by gender, mean age, resource used and planned CfE computing science level. School B has three classes P4, P5 and P6 with 24, 19 and 26 children respectively. School M has three classes P3, P4 and P5 with 21, 29 and 28 children respectively. The mean age of all participants in each group shows that the children in School B class P3 mean of 6.1 is almost one year younger than the mean of children in School M class P3 with a mean age of 7 and P4 and P5.

School	Class	Code	Ambassador Gender		Number of Children	Children Gender		Age Mean	CfE CS*	Resource
			F	M		F	M			
B	P3	B		1	24	10	14	6.1	1 st level	Scratch
	P4	B		1	19	8	11	7.1		
	P5	B		1	26	14	15	8.1		
M	P3	M	1		21	7	14	7	1 st level	Scratch
	P4	M	1		29	11	18	8.2		
	P5	M	1		28	15	13	9		
F= Female M= Male										

Table 29 Categorisation of participants in the study

7.2.6 Materials

The Ambassadors plan three CS lessons based on the Scottish CS curriculum framework. This curriculum is research informed with individual learning outcomes called experiences and outcomes grouped into three themes or organisers; (1) core computing science concepts; (2) how tools and languages use the concepts; (3) application of learning by creating solutions. Step 2 is the bridge between step one – processes in the real world and step 3 virtual representations of real-life situations. The Scottish curriculum approach is explained in more detail later on in the chapter.

A summary of the learning is below:

Lesson One Step 1 (Overview):

LI1.1, LI1.2 SC1.2 SC1.3

- Part 1. What do you know about computers? What is a process? Hand Jive activity.

- Part 2.: Instruction station - green and red flag; start and stop processes. Parallel and sequential processes identified in real world and Rube Goldberg Machine video
- Part 3. Break down a complex problem (making breakfast)
- Part 4. Program the teacher, optimise. Revisit hand jive and optimise.

Lesson Two Step 2 (Overview):

LI2.1 LI2.2 SC2.1 SC2.2 SC 2.3 SC 2.4

- Part 1. Introduction to ScratchJr (explaining what each ScratchJr tile does) and quiz on tiles
- Part 2. Predict ScratchJr scripts
- Part 3. Program the Teacher using ScratchJr tiles

Lesson Three Step 3 (Overview):

3.1, 3.2, .3.3 SC: SC3.1, SC3.2, SC3.3

- Part 1. Warm-up (Recap) + Prediction quiz (Show 4 print screens of ScratchJr scripts from SAL2 and ask the pupils to predict where the character is going to end up at)
- Part 2. Optimisation. Optimise a game about reaching a cake. Pair up the pupils and explain how to load and start the project. Extend a game about a basketball cat. Show the pupils how it should look like on the screen. The pupils should extend the project so that the cat throws the ball to the basketball hoop and scores a point.

The lesson plans align with the CfE CS curricula at 1st level (age 5.5 to 8.5); ambassadors take account of the three step approach with children learning about CS concepts in real life situations before moving to understanding the tools and how they use these concepts before building.

Children's views on their learning are gathered through KWL grids. KWL grids were first developed as a teaching strategy in the USA by Ogle (1986,1989) and Carrad (1987). Although not used as a research tool within the CSED research community, the grid structure establishes what children know about a subject (K) (column one), what they want to know (W) (column two), and at the end of a series of lessons they can record what they learned (L) (column three), (Greenwood, R., 2019). The children in this study are familiar with this process through learning in other areas of the curricula. Typically, a primary teacher uses this approach to support children to lead their own learning and complement planned learning. The first column provides clarity on children's prior knowledge. The final column shows the depth of learning in the course.

The third column is completed at the end of the study.

What I know about CS	What I want to know	What I learned

Table 30 KWL grid headings

7.2.7 Procedure

The Ambassadors collaborate on the planned learning for the study. Materials for delivery which take account of the CfE three step (comprehension-first) approach which have been moderated by a CSED expert and primary education expert. Each Ambassador delivers 3 x 1 hour lessons to three classes of different age groups (age 7-10) (Appendix J). Before the lesson start, children complete the KWL grids and the online survey about their play interests.

Lessons are delivered independently of the class teacher. All lessons are led mainly from the front of the class. Children work in small groups. The class teacher and a classroom assistant attend each session. The school staff attending provide support to children and moderation for the assessment. The study deliberately uses the same materials for each of the year groups due to no prior knowledge of the participants. Scaffolding techniques would be used if needed.

Each lesson has a learning intention to focus the teaching in line with the CfE CS curricula. (LI). The lesson shows success criteria for children to self-assess and involve them in their learning.

7.2.8 Data

Data collection

Qualitative and quantitative data gathered from the study provides evidence for the findings.

Qualitative and quantitative data analysis

The study consists of two bodies of work. Firstly, an exploration of children’s success rates during an introductory programming course delivered by CS experts with no primary teaching experience. Secondly, it analyses children’s success in the course including age, gender and early childhood experiences outside formal education. The latter is measured through children’s attitudes to toys, games and activities. The study gathered qualitative

and quantitative data. The following section explains how some of the data was quantified for analysis.

The range of data sources include qualitative and quantitative and can be viewed below.

Qualitative data & Analysis	Quantitative data and Analysis
Ambassadors lesson plans	Age
Ambassador's journals	Gender
Children's self-assessment	Toys, games and activities enjoyment
Children's KWL	Success in lesson
Scottish CS curriculum	

Table 31 Sources of qualitative and quantitative data

Data sources are as follows, with an overview in Figure 3. Convergent parallel design methodology (Edmonds and Kennedy, 2017) is used to collect qualitative and quantitative data simultaneously and independently. 2 Ambassadors, 6 classes, 158 children age 7-10 participate across the two schools. A convergent parallel mixed methods approach

The Ambassadors and children's reflections alongside quantitative data of known predictors of success: Age; gender and interests in activities outside formal learning are examined. In the analysis stage equal weighting is given to the quantitative and qualitative data. The results are then compared looking for patterns or contradictions. This study focuses on 4 factors contributing to their progress and motivation in learning CS concepts. Four themes for reporting findings in this study are, the role of the educator leading the learning (the Ambassador), the children's learning experience and the CS curricula (Concepts).

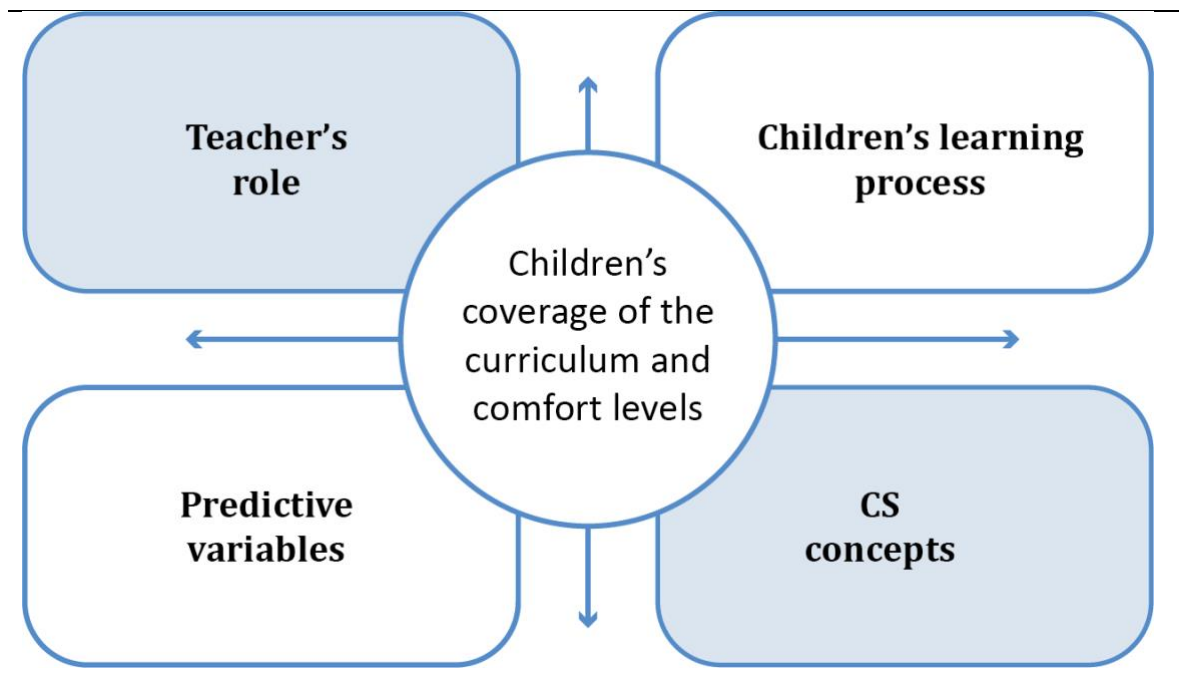


Figure 8 Four study themes

The figure above illustrates each of the four bodies of work (themes) within the study and their interconnectedness with the children's progress and levels of motivation.

The study looks at the well documented predictor variables of age and gender. STEM research (Ferrara, Hirsh-Pasek, et al., 2011) shows a link with engineering and early block play; therefore, the study considers the role of children's play interests more broadly and its impact on children's progress in the course. The complex data gathering involved in the study is well-suited to a convergent parallel mixed method approach.

Once analysed the results for the qualitative and quantitative data will be presented separately under the three themes of the Ambassador's role, Learners experiences, CS concepts. The results are subsequently merged for comparison.

Qualitative data collection and analysis

Ambassador's role

- Ambassador's reflections in their journals. The Ambassadors write reflections on the lessons delivered to the children. The reflections show how each lesson has gone from the perspective of the Ambassador and any points of interest about the children's learning.

Children's learning through KWL grids. KWL grids are explained in section 6.2.6. The goal of these grids is to establish children's prior CS knowledge and their expectations of the course. These grids enable a comparison to be made about what children learn and what they wanted to learn.

- Children's knowledge prior to starting the introductory programming course about what they know and want to know (**K**).
- Children's knowledge about what they learned overall (**W**).
- Children's learning lesson by lesson (**L**).

Children's learning through self-assessment templates

- The children assess their progress using self-assessment templates that they are familiar with in other curricula areas of their learning. An open question asks children to note any comments about their learning. The templates enable the children to assess their learning using a colour coding system which becomes a quantifiable measure used in the quantitative section of the study. 'Red' if they found the task very difficult, 'Amber' if they needed some help, 'Green' if they managed the task without support. The colours were transferred into points for the data analysis.

CS Concepts

- Ambassador's teaching plans linked to the CfE curricula (Appendix 4)

Predictor variables

- N/A

Quantitative data collection and analysis

Each lesson has learning outcomes with clear learning intentions (LI) related to the Scottish Curriculum philosophy of the three-step approach outlined above. The success of the lesson is measured in criteria specified during the planning stages of the lessons. Children evaluate their own learning using this success criteria (SC). Table 32 shows success criteria for each lesson. Children evaluate their own learning of each of the success criteria (SC) using Red, Amber or Green colours and the teacher moderates their assessment. These responses are quantifiable and a score of 1 is awarded to red, a score of 2 is awarded to amber and a score of 3 is awarded to green.

LESSON ONE SC	LESSON TWO SC	LESSON THREE SC
SC1.1 I can carry out the steps of a process description as they are given to me.	SC2.1 I know the meaning of some Scratch Jr blocks	SC3.1 I can read and understand a basic Scratch Jnr script.
SC1.2 I can create and improve a process description	SC2.2 I can read and understand a basic Scratch Jr Script	SC 3.2 I can predict the outcomes of a basic Scratch Jr script.
	ISC2.3 can predict the outcomes of a basic Scratch Jr Script	SC3.3 I can optimise a basic script
	SC 2.4 I can create my own sequence of instructions using basic Scratch Jnr tiles.	

Table 32 Shows the success criteria children use for self-assessment.

Data collection and scoring of children’s self-assessment

Children’s self-assessment scoring Across the three lessons in school, children’s progress is measured using each lesson’s success criteria. Children in the study are familiar with self-assessment approaches and the definition of using red, amber or green. Scoring max 3 points for each success criteria.

- G (green light) – “I understand this very well” = a score of 3
- A (amber light) – “I need a bit of support but understand the basics” = a score of 2 points
- R (red light) – help “I don’t understand” = a score of 1

Each child can achieve a maximum of 27 and a minimum score of 9. Children self-assess throughout each lesson and assessments are moderated by the class teachers. A sample of a full class score is shown in Appendix I and children’s self-assessment in the table below.

SUCCESS CRITERIA – children’s self-assessment scoring

CHILD	LESSON ONE		LESSON TWO				LESSON THREE			Total
	SC1.1	SC1.2	SC2.1	SC2.2	SC2.3	SC2.4	SC3.1	SC3.2	SC3.3	
B4	A	G	G	G	G	G	G	R	G	24
B45	A	G	G	G	G	G	G	G	G	26
B6	A	G	G	R	R	G	G	A	G	21
B7	A	G	G	A	A	G	A	A	A	21
RED = 1 POINT, AMBER = 2 POINTS, GREEN = 3 POINTS										

Table 33 Sample self-assessment

If children colour the traffic light in their self-assessment sheet R (red) they receive a score of 1, A (amber) they receive a score of 2, and G (green) they receive a score of 3.

Children in this study are familiar with this assessment process. A summative assessment gathering qualitative responses took place at the end of the series of lessons using the KWL grids. These statements were not quantifiable.

Survey responses collapsed by play category

The study considers predictors of CS success and known variables such as age and gender. An added dimension is extending the knowledge that block play develops engineering skills in young children to explore a relationship with play interest outside formal education more broadly and success. A questionnaire (Cutts et al., 2017) aimed to gather information about children’s play interests is categorised. Whitbread’s five areas of play provide the framing for the 37 play activities in the questionnaire are compressed to Whitbread’s areas of physical (PHY), pretence (Pre), object (Obj), symbolic(Sym), construction (Con) and games with rules(Game) mentioned previously. Children score their enjoyment levels from N/A (not appropriate or no experience) to 5. An open text box within the questionnaire provides space for gathering qualitative information about the play interests not covered by the questionnaire. An additional digital toys (DT) category was added to reflect the wide range of toys children access. The questionnaire

categorisation is moderated by four early years experts for professional views on the validity of its theoretical framework.

Children’s enjoyment levels toys, games and activities

Children’s individual survey responses collapsed to each of the play categories

PLAY TOYS GAMES AND ACTIVITIES SURVEY QUESTIONS			
THEME			
PHY		Climbing	Wrestling
		Riding bikes	Chasing friends
		Playing with balls	Physical play
		Trampolines	Dancing
OBJ		Building models	Play with stones, sticks, mud
		Sand play	Play with objects
		Water play	Lego
SYM		Painting/drawing	Singing
		Reading on my own	Music
		Drawing	Play doh
		Being read to by others	Writing
PRE		Dolls	Pretend shops
		Superheroes	Playing at schools
		Dressing up	Pretend play
GAME		Card games	Team games
		Games with rules	Guessing games
		Board games	Playground games
DT		Computer games	
		Electronic games	

Table 34 Survey questions organised by play categories.

A correlation with children’s progress score and age, gender and enjoyment levels of play categories results is undertaken.

7.3 Results

7.3.1 Thematic analysis

The thematic analysis of the Ambassador's and children's qualitative reflections provides insights into the design and planned learning that can be compared and contrasted with quantitative data gathered. It shows children's experiences, prior knowledge and expectations for the course. The quantitative data shows children's progress through the CS concepts and predictors of success. The results will be presented using the themes outlined in the introduction.

The ambassadors wrote reflections on each lesson. The reflections are unstructured although covered main points arising from the lessons. The Ambassadors wrote approximately three pages of text and excerpts were manually extracted and coded using NVIVO.

At the start of the study all children wrote reflections on what they know about computing science (know) what they wanted to learn (learn). They reflected on learning lesson by lesson and a final reflection on the learning one week after lesson three, the final lesson was delivered. Most children wrote a sentence about their learning. The majority made two points. The average response was 12-13 words per lesson. Across all the reflections, less than half made no response.

Patterns from the data were identified using qualitative methods. Each response was manually transcribed into an excel spreadsheet in order to become familiar with the data. The thematic analysis on the ambassadors and children's reflections generate codes. These codes are sorted in themes and dimensions collapsing highly similar codes. Next themes were revised and refined. Each excerpt is coded.

A standard interrater reliability methodology is employed. Two individuals not involved in the study and no knowledge of computing science education independently coded the work. After the samples were scored the lead researcher and the independent individuals calculated the inter-reliability score. The ratings resulted in an inter-related reliability score kappa of .85 indicating substantial agreement.

The thematic hierarchy analysis children and Ambassadors comments is organised as follows. Both the Ambassadors and children's reflections are categorised under the learning dimension.

Children's learning lesson by lesson (Lesson level) hierarchy codes are recorded. Comments show the level of challenge experienced by children and the activities that they enjoyed.

The Ambassador's reflections and were grouped together yielding themes about learning as follows:

The Ambassadors reflections

- Children's learning
- Classroom management

A separate analysis of what children know, want to know and learned (KWL) in their KWL grids is undertaken. The analysis focuses on the most prominent themes in each domain by discussing the related sub theme.

7.3.2 The Ambassador’s reflections

Table 35 shows the hierarchy of codes mentioned above arising from the thematic analysis of the Ambassadors and children’s ongoing reflections on the learning throughout the course. The reflection coding collapses twenty-five codes to the three dimensions of:

- (1) Learning – The activity itself, how children learn, and the level of challenge observed.
- (2) CS concepts – The processes observed, unplugged learning and Scratch tools
- (3) Attitudes- Levels of motivation and self-worth including growth mindset

Hierarchy of code from children’s reflections at Lesson Level

Revised Codes	Themes	Dimensions
Groups	How	Learning
Easy	Challenge	
Hard		
Instructions	Activity	
Selfies		
Making games		
Dance		

Table 35 Hierarchy of children’s reflections thematic codes. Learning dimension

Revised Codes	Themes	Dimensions
Parallel	Processes	CS Concepts
Sequential		
Programming	Unplugged	
Optimising		
Relationship with programming		
Scratch	Tools	
Reading tiles		
Tiles actions		
Creating scripts		
Motivation	Behaviours	Attitudes to learning
Self-worth		
Growth mindset		

Table 36 Hierarchy of thematic analysis codes CS concepts and Attitudes to learning.

7.3.3 **KWL themes**

The KWL reflections collapse twenty-six codes to two dimensions

- (1) use of technology
- (2) Computing science Discipline

The two dimensions are informed by the five themes of

1. Software
2. Hardware/Networks
3. Programming
4. Processes
5. Computers in society.

Revised Codes	Themes	Dimensions
Word processing	Software	IT Use
Security		
Search engines		
Research		
Application		
Online shopping		
Peripherals	Hardware/Networks	CS Discipline
How they work		
How to build		
Operate		
Internet		
Wifi		
Electronics		
Scratch	Programming	
How to make a game		
Fix software		
Make a website		
Hack		
Parallel	Processes	
Sequential		
Processes		
Importance	Computers in society	
Invented		
Entrepreneurial		

Table 37 The hierarchy of KWL codes

7.3.4 **Ambassador's lesson plans**

7.3.5 **The Ambassadors' journals**

The Ambassador's views of each lesson were analysed for deeper insights. The focus for the analysis is primarily on children's learning, however, a high level of motivation and engagement is reported by the children and triangulated by the Ambassadors,

“Overall, in all lessons of the classes attitudes from both teachers and students were very positive. The children were excited to try the activities” **Ambassador M**

The Ambassadors were clear from the outset of the lesson structure and theoretical underpinning.

“kids need a link between understanding the world through computational thinking and, understanding the Computing Technology and designing great things themselves; If the connection is not clear – tweak the workshops so that pupils can see it more easily. We have setup our lesson trilogy in a way that the second lecture would serve as the core link between physical and virtual worlds” **Ambassador B**

They included activities that helped children to visualise the concepts. Talking about the activities final year CS Ambassador:

“I think it really showed children the concepts in the real world and it made them think and visualise the moves”

When describing approaches to reinforce and consolidate learning:

“I use examples from subjects that the pupils have already studied so they will remember the new concepts better. For example, parallel and sequential (in maths parallel lines, sequences) I tried this with P3 and P4 it worked.”

When introducing the Scratch Jr tile commands to program the ‘Robo Teacher’ they recapped on the underlying concepts.

“When I was explaining the commands, I made a recap to the previous lesson i.e.: parallel and sequential processes and I was glad they were remembering the theory”

Ambassador B

Both Ambassadors commented on the learning intentions as more appropriate for the children at P5.

“to my mind, this workshop suited best P5 as they understand the concepts better”

Ambassador M

“I have noticed a huge difference in the performance of P3 and P5. While P3 doing a lot of mistakes, P5 managed to do it with almost none. I also noticed that in P5, they were able to alter their instructions during the time I was executing them, while P3 could not do that”

Ambassador M

And this point is reinforced in relation to the P3 learning.

“P3 on the other hand were a bit lost.”

Ambassador M

7.3.6 Classroom management

Both Ambassadors commented on barriers to complete the planned learning. This ranged from issues with laptops, resources for the materials and the children’s age with the P5 children achieving more.

“as a conclusion, everything went well apart from some technical problems”

Ambassador B

“At some point, there were just too many students asking me to see that they are ready or that they are struggling and being only me (since the teacher have not used ScratchJr) with 33 pupils was just impossible.” **Ambassador M.**

7.3.7 Children’s learning experiences

The class teacher for each class observed the lessons and moderated the children’s self-assessment of their learning. The class teacher had no input into planning. Children self-assessed their progress during the lessons within their learning journal. The children are familiar with this approach. A KWL grid is used to provide insight into perceived gaps in knowledge or expectations of a computer science course in the primary school. Used

alongside the learning, reflections a deeper understanding of children's prior knowledge, interests, expectations of the course and overall learning is shown. The text was extracted using NVIVO and the initial codes auto generated.

Two themes emerged from the children's KWL reflections

1. USE OF TECHNOLOGY
2. COMPUTING SCIENCE DISCIPLINE

Children's learning at lesson level

Summary: The KWL grid and children's reflections at lesson level show children have little or no prior CS knowledge. The course content delivered did not relate to the children's expectations of CS or what they wanted to know. Lesson by lesson, there is a match in children's overall learning and the learning intentions of the planned session. Children reflect positively on their enjoyment levels of the activities.

Children's learning at lesson level Children's learning at lesson level within the **Attitudes to learning dimension** includes themes of children's levels of motivation, enjoyment and self-worth. Written reflections show children are consistently very positive about the learning activity in which they are involved and in the progress that they make. **Motivation:** This refers to children's levels of willingness to do the activity. Enjoyment levels are high across the responses and although not every child reported that their levels of motivation or enjoyment, no child indicated that they did not enjoy their learning

“It was amazing learning parallel processing, and it was awesome learning sequential.” *School B P3 child*

And

“I found the lesson interesting, fun and awesome. I learned what palal (sic parallel) processing is.” *School B P5 child*

and

“It was really interesting and very fun. I can't wait for next week.” *School M P4 child*

As the lessons progress, while children continue to enjoy their learning there is an increased focus on the learning outcomes of the lesson and less use of the enthusiastic vocabulary such as 'awesome' and 'exciting'

“I followed instructions using the blocks it was fun.” And “I think it was fun and I could predict well.” *School B P5 child*

The **Learning dimension** themes are those involved in the learning activities designed to promote learning. They do not include the learning outcome of the activity but the activity itself and a child’s perceived level of challenge.

The **Challenge** theme refers to children’s view of the lesson’s level of difficulty. While some of the children found the lesson difficult or confusing, they qualify the statements with high levels of enjoyment.

“This lesson was interesting and fun. I enjoyed it It was Konfusing (sic confusing).”

School B P4 child

And where a particular element was challenging the children reflected on how they would improve this through more practice.

“Some of it was hard and I need to work on guessing where the haracter (sic character) will go.”

School M P4 child

The **Activity** theme includes the event or task that children undertake to gain the planned learning. Most children reflect on the activity itself at surface level and very few make connections with the activity and its learning intention.

“I learned what process means. I liked the cool videos”

School M P5 child

One child making a connection between the activity and learning wrote:

“I learned about programming. I learned how to break big problems into small instructions”

School B P4 child

How Tools and Language use the concepts. This theme focuses on the visual programming environment Scratch and children's reading, writing and understanding of the associated tools. The lessons focus on the relationship between the tangible real-life processes towards the necessary computer tools for programming. Across all schools most children wrote about Scratch at a surface description. With a very small number making the link between the real world and the virtual world.

“I leaned (sic learned) how to use Scratch; I thought that it was awesome. I've learned how to use the little blocks”

School M P5 child

However, a few children made the link.

“I learned that Scratch tiles are processes that make computers work”

School B P6 child

and

“I learned about using processes to code via Scratch”

School M P4 child

and

“I know what a parallel process is on a Scratch program.”

School B P4 child

Before introducing the tiles, the Ambassador asked children to guess what they mean. This was expected to be a simple introduction because Scratch Jr is aimed at very young children. However, there were a few errors with responses, in particular with 50% P3 children, 50% P4 and P5 children knowing the move right tile. Reflection confirmed this observation.

“It was fun and I guessed all the blocks.”

School B P6 child

While others found it challenging.

“It was fun. I did not guess what some of the blocks were”

School M P6 child

Some distinguished reading and writing and how they used Scratch.

KWL Analysis

Children record what they know and what they want to know about a themes or topic before starting the planned learning. Approximately one week after the session is complete each child reviews their learning and compares this with the expected outcomes of the class teacher.

Analysis of children’s prior knowledge, what they want to learn and what they learned over the three lessons resulted in two domains. **IT Use** and the **CS** discipline with associated subthemes. The following section focuses on the most prominent themes and subthemes concluding with a comparison of children’s learning overall and the introductory programming course aims.

2.3 Use of IT dimension: The themes in the use of IT dimension are those that involve using technology and applications when describing prior knowledge most children made reference to application. Children report that prior to the introductory programming lessons they know:

“How to message people via email and text; I know how to use Microsoft office. I know how to type on a keyboard. You can shop on a computer for food.”

School B P5 child

And

“I can use a mouse. How to research using the internet. How to play computer games. Download apps.”

School B P5 child

And

“I can use Microsoft office well. I use messages to text my friend. I can use my voice to go on apps. I can shop online. I can make a website using Wix.Com How to upload a YouTube video. How to use the internet.”

School B P5 child

Computing Science Discipline: The themes in CS Discipline domain involve CS as a whole and any aspects of computers and computing that may be studied in a CS curriculum at any level.

Hardware/networks: This theme refers to computer and network designs, it includes building or “fixing” computers and networks. Most students wanted to learn about the physical aspect of computers and networks, how to build, fix them and how they work.

“What are all the bits inside a computer. How does it work” and “How are computers powered?”

School M P6 child

And

“How to hack a computer?”

School B P5 child

“How does wifi work?”

School M P5 child

“What computers are in space rockets to make them go?”

School M P5 child

“How apps work. How to make apps. How does the internet work?”

School B P5 child

Programming theme

“How to code a robot not a screen one.”

School M P3 child

One child had a deeper knowledge and wrote that he knew:

“How to build a PC How to know java coding, how to set up 2006 macmini. I know how to install lynx on a chrome book. What I create a new account on. How to make a website using java script. How to use a VPN. How to set up screen sharing and file sharing.”

School B P6 child

This was confirmed by the class teacher although the child’s behaviours to those of the class peers was observed and captured by the Ambassadors in the learning journal reflections.

The same child did not engage in the dance and was bemused by the enthusiasm of the class.

“However, I have noticed something very interesting in his behaviour – during one of the activities (watching and repeating a hand jive), which every single student in the class but him enjoyed very much, he seemed very confused why the others are so excited to do it. When the class was repeating the hand jive, he was standing up, just like everyone else, but he did not make any single movement. I could tell from his reaction and body language that he found it rather stupid. I cannot wait to see him doing the coding exercise.” **Ambassador M**

Computers in society: this theme refers to the history of computers, their importance and entrepreneurial opportunities.

A few wanted to know who invented computers, how many there are in Glasgow, the world.

“how many computers are in banks”

School B P5 child

One child took this further and asked what type of computers were used in space rockets,

A few recognised the entrepreneurial opportunities and asked:

“How to make games. How to make money for free”

School B P5 child

High numbers of children report that they have no prior knowledge:

“no don't know sorry”

School B P5 child

and

“I have learned to work a iPad. I know how to control my character. I have learned how to use Scratch Jr. I know what a sequential process is.”

School B P5 child

7.3.8 CS Concepts Results

Each lesson has a learning intention (LI) which the Ambassadors use to focus the lesson activity. To measure the success of the planned learning the Ambassador students share with the children the success criteria (SC) for each lesson. Children self-assess their progress against the success criteria. Each lesson learning intention and success criteria links directly to the Scottish CfE CS curricula using a code in the final column.

CS 0.1 = Computing science early level 1st organiser CS concepts.

CSC 2.1 = Computing science curricula 2nd level 1st organiser CS concepts.

TL 0.1 = 2nd organiser tools for learning early level.

AOL 0.1= is application of learning 3rd organiser early level 1st statement.

Lesson 1 code	Learning Intention (LI) Success Criteria (SC)	CfE Code
LI 1.1	To introduce computational concepts of processes and process descriptions	
LI 1.2	To apply computational concepts in familiar contexts.	
SC 1.1	I can carry out the steps of a process description as they are given to me.	CSC0.1
SC 1.2	I can create and improve a process description	CSC 2.1
Lesson 2 code	Learning Intention (LI) Success Criteria (SC)	CFE Code
LI 2.1	To apply computational concepts in unfamiliar contexts.	
LI 2.2	To introduce computational concepts using programming language.	
SC 2.1	I know the meaning of some Scratch Jr blocks	CS0.1
SC 2.2	I can read and understand a basic Scratch Jr Script	TL1.1
SC 2.3	I can predict the outcomes of a basic Scratch Jr Script	TL0.2,2.2
SC 2.4	I can create my own sequence of instructions using basic Scratch Jnr tiles.	AOL0.1
Lesson 3 code	Learning Intention (LI) Success Criteria (SC)	CFE Code
LI 3.1	To use programming tools to apply computational concepts.	
LI 3.2	To understand basic scripts and predict outcomes.	
LI 3.3	To modify and create basic scripts.	
SC 3.1	I can read and understand a basic Scratch Jr script.	TL1.2,2.2
SC 3.2	I can predict the outcomes of a basic Scratch Jr script.	TL1.2,2.2
SC 3.3	I can optimise a basic Scratch Jr script	AOL1.5,2.3
	CS= CS concepts TL = Tools for learning AOL = Application of learning	

Table 38 Lesson learning intention and success criteria codes

7.3.9 Progress through the Curriculum

Children's self-assessment and statements in their KWL grids show that they focus on the activity. A few n=3 make the link between the unplugged activity and how the programs work. Most describe the programming tool with few reporting that they can read and write scripts using ScratchJr. A few children report that making predictions about unplugged activities and making predictions about the behaviour of a script is challenging.

The KWL focus shows almost all children's prior knowledge about CS is with the use of technology. Small numbers used Scratch previously. Most children wanted to know how to fix computers, build computers, how wifi worked and few wanted to learn programming. Comparisons between children's learning at lesson level and learning overall show most focus on the Scratch environment and tools.

Moving to the second dimension "CS concepts": This theme relates to the CS concepts and the core learning intention of the introductory programming course. Most children reflect on processes using technically accurate vocabulary, although the statements are at surface level.

"I found the lesson interesting, fun and awesome. I learned what parallel processing.

School B P5 child

And

"I enjoyed the dance. I learned a new word parallel"

School B P5 child

and

"It was amazing learning parallel processing, and it was awesome learning sequential"

School B P5 child

and

"I learnt what a sequential processing and parallel processing is and it was so so so fun"

School B P5 child

A few children expand on the surface use of the process vocabulary using statements reflecting showing and accurate understanding of the underlying concepts.

"I learned that sequential means you do one then then another"

School B P5 child

and

“Every second you are processing different things”

School B P5 child

and

“I learned that walking is a process”

School B P5 child

Unplugged theme: This includes the activities and learning that teach children about computer science concepts without computers. The unplugged theme had clusters of reflections across the three classes and lessons. Children are physically active in these activities although acquisition of the underlying learning is less clear in the reflections. Children wrote about programming the teacher and each one mentioned high levels of motivation.

“I enjoyed giving instructions. I learned that computer science is about giving instructions”

School B P5 child

“I enjoyed giving the instructions. I learned that paralelle (sic parallel) means that you do something at the same time”

School B P5 child

Using an unplugged approach to learn optimisation worked well for one child to visualise errors and make improvements.

“I liked giving x (sic the teacher) because if I made a mistake, I could see it.”

School B P5 child

And

“I know about processes and this is how computers work”

School B P5 child

Embedded within the lessons was the use of prediction and quantitative scores show marginal differences when requiring prediction. This was reflected in the learning of children n=4 found this challenging.

“I think I need to practice predictions.”

School B P5 child

And

“Some of it was hard and I need to work on guessing where the haracter (sic character) will go.”

School B P5 child

7.3.10 Measuring children's Total scores across the three lessons:

The table below shows that the scores for the whole course range from 18 to 27 with a mean of 24.. Most children score 21-24 with 2 children achieving full marks. The two children achieving full marks are boys. There is a relatively normal distribution and further analysis confirmed that there are no outliers.

Question by question scores: All success criteria scores are broadly similar.

	Min	Max	Mean
Children's Total Success Criteria score from all lessons	18	27	24.4
Children's total Success Criteria score for lesson one	3	6	5
Children's total Success Criteria score for lesson two	8	12	11.1
Children's total Success Criteria score for lesson three	3	9	8.4

Table 39 Mean and scores across cohort

Year group	P3M	P3B	P4M	P4B	P5M	P5B	Total
TOTAL	24.4	22.8	25.7	23.3	25.7	24.2	146.1
SUCCESS CRITERIA SCORES (SC)							

Table 40 Total and question level scores (mean)

Year group	P3M	P3B	P4M	P4B	P5M	P5B	Total
LESSON ONE SC TOTAL	5.1	4.4	5.7	5.0	5.7	4.7	30.6
SC1.1 I can carry out the steps of a process description as they are given to me.	2.2	2.0	2.8	2.0	2.8	2.0	13.8
SC1.2 I can create and improve a process description	2.9	2.4	2.9	3.0	2.9	2.7	16.8
LESSON Two SC TOTAL	5.0	4.3	5.7	4.9	5.7	4.7	30.3
SC2.1 I know the meaning of some Scratch Jr blocks	2.9	2.6	2.9	2.8	2.9	2.8	16.9
SC2.2 I can read and understand a basic Scratch Jr Script	2.7	2.6	2.7	2.8	2.7	2.7	16.2
I SC2.3 can predict the outcomes of a basic Scratch Jr Script	2.5	2.6	2.7	2.5	2.7	2.7	15.7
SC 2.4 I can create my own sequence of instructions using basic Scratch Jnr tiles.	2.8	2.5	2.9	2.8	2.9	2.8	16.7
LESSON THREE SC TOTAL	10.8	10.2	11.3	10.9	11.3	11.1	65.6
SC3.1 I can read and understand a basic Scratch Jnr script.	2.9	2.7	2.9	2.7	2.9	2.8	16.9

SC 3.2 I can predict the outcomes of a basic Scratch Jr script.	2.6	2.7	2.9	2.2	2.9	2.8	16.1
SC3.3 I can optimise a basic script	3	2.8	2.9	2.5	2.9	2.9	17
Total Scores SC for lesson three	8.5	8.2	8.7	7.4	8.7	8.4	49.9

Table 41 Question level scores (mean) by each lesson

7.3.11 Total scores across the three lessons by class:

At course level the results show in descending order that P5M scores are higher with a total mean score of 35.7 and P3B score the lowest with a total mean score of 22.8 for each lesson. The following table shows the scores for the whole course in descending order

Class	P5M	P4M	P3M	P5B	P4B	P3B
Total score for introductory programming course	25.7	25.7	24.4	24.2	23.2	22.8

Table 42 Mean performance for each class

At lesson level, P5M score the highest in each lesson with P3B the lowest in two out of the three lessons. The lessons are calculated individually and therefore the scores do not roll from one lesson to the next.

Class	P5M	P4M	P3M	P4B	P5B	P3B
Total mean score Lesson 1	5.7	5.7	5	4.9	4.7	4.3

Table 43 Mean performance for lesson one

Class	P5M	P4M	P5B	P4B	P3M	P3B
Total mean score Lesson 2	11.3	11.3	11.1	10.9	10.9	10.3

Table 44 Mean performance lesson two

Class	P5M	P4M	P3M	P5B	P3B	P4B
Total mean score Lesson 3	8.7	8.7	8.5	8.4	8.2	7.4

Table 45 Mean performance lesson three

Scores show optimising the Scratch script, reading and understanding basic scripts have the highest scores. The lowest scores are for carrying out processes in lesson one and predicting the outcomes of a basic Scratch script.

7.3.12 Children’s enjoyment levels of play

The preliminary analysis involves data cleaning by removing participants with missing data. The 158 responses to each of the survey questions were initially collapsed into each of Whitbread’s play categories with a grouping of digital toys and games and construction toys added. These categories are shown in Table 47. Each child’s response was analysed using SPSS descriptive statistics to establish any patterns across the class sample and their play interests. The descriptive data provides characteristics of the sample and the Skewness and Kurtosis provides indication of the symmetry of the data distribution. For construction it would appear that most of the distribution is within the tails of the curve.

Play categories.	Range of response from lowest to highest.		Distribution of the data		
	Minimum	Maximum	Mean	Skewness	Kurtosis
The mean scores of each play type for the cohort (158) children					
Phy	0.50	5.00	3.1683	0.88055	-0.164
Obj	1.00	5.00	3.2756	0.90343	0.318
Con	0.00	5.00	3.3833	1.61309	3.362
Sym	0.78	5.00	3.0881	0.90270	-0.076
Pre	0.00	5.00	2.9833	1.05793	-0.379
Games	0.50	5.00	3.2178	0.98079	-0.500
DT	0.00	5.00	3.2000	1.36593	-0.464

Table 46 Children’s responses in the enjoyment survey showing by play category

CORRELATION

- Correlation of children’s scores and attitudes to play categories
- Correlation of children’s age, gender and attitudes to play categories

Pearson’s correlation was used to explore the strength of relationship as one variable increases.

	AGE	Gend	total	Phys	Obj	Con	Sym	Pre	Game	DT
Age	1.0	-0.1	.48**	-.30**	-0.1	-0.1	-0.1	-.18*	-.34**	0.1
Gend	-0.1	1.0	-0.1	0.1	-0.1	0.0	-0.1	-.21**	0.0	0.0
Tot SC	.48**	-0.1	1.0	-.23**	0.0	0.1	0.0	-0.1	-.23**	0.0
Phys	-.31**	0.1	-.23**	1.0	.37**	0.1	.42**	.44**	.52**	.34**
Obj	-0.1	-0.1	0.0	.37**	1.0	.78**	.42**	.40**	.41**	.21**
Con	-0.1	0.0	0.1	0.1	.78**	1.0	0.2	0.1	.25**	0.1
Sym	-0.1	-0.1	0.0	.42**	.42**	0.2	1.0	.56**	.56**	.24**
Pre	-.18*	-.21**	-0.1	.44**	.40**	0.1	.56**	1.0	.48**	.24**
Game	-.34**	0.0	-.231**	.52**	.41**	.25**	.56**	.48**	1.0	.32**
Dig	0.1	0.0	0.0	.34**	.21**	0.1	.24**	.24**	.32**	1.0

Table 47 Correlation of toys games and age, gender total scores showing **

The results show a relationship with children’s age and their total score. This reflects the view of the Ambassadors M but not the total results in descending order where children in school M perform overall better than children in school B.

Children with lower levels of enjoyment in physical play and games with rules score more than those with high levels of interests in these areas. There is no statistically significant relationship with children’s interests in each of the play areas and the total scores in the course. There is a statistical significance between groups of play. Pretence, physical, object, symbolic, games with rules and digital are statistically significant. The cohort has

an enjoyment level across a breadth of experiences which aligns with the schools set in an area of high socioeconomic status.

7.4 Comparison of qualitative and quantitative results.

The comparison of the qualitative and quantitative data leads to convergence and divergence interpretation. The qualitative and quantitative comparison is structured as follows

- Whole course analyses
- Lesson by lesson analysis
- Question level analysis.

7.4.1 Whole course level analysis

Lesson plans show Ambassador’s planned the same learning for each class. However, there was sufficient content within the materials to enable progression because no child had prior programming experiences. Interestingly, in both schools the older children achieve more than the younger children although the margins are small. The Ambassadors in school M reports that the materials are more suitable for the older children and this statement cannot be generalised across the two cohorts because the youngest children in school M perform more than the eldest in school B. However, across the cohort, the whole class mean quantitative scores across the introductory programming course shows children in school M achieve marginally more than children in school B irrespective of their age. The observation by Ambassador M is supported by the data for school M. However, it cannot be generalised across the cohort and it could be due to teacher expectation that older children make more progress. The KWL grids confirm that none of the children have prior CS experience in programming, they all received the same materials and this discrepancy need consideration.

Class	P5M	P4M	P3M	P5B	P4B	P3B
Mean score for whole introductory programming course	25.7	25.7	24.4	24.2	23.2	22.8

Table 48 Mean progress of each class

7.4.2 Lesson level

The Ambassadors' learning plans show increasing complexity in each of the lessons. Children progress from observing processes in real life to more abstract tasks where they need to understand how the computational tools use the concepts and finally using the computational tools to create a solution. The table below shows in descending order class performance with P5M scoring the highest in lesson 1, lesson two and lesson three.

Lesson 1	P5M	P4M	P3M	P4B	P5B	P3B
Lesson 2	P5M	P4M	P5B	P4B	P3M	P3B
Lesson 3	P5M	P4M	P3M	P5B	P3B	P4B

Table 49 Class progress lesson by lesson in descending order.

In lesson one and lesson two, the youngest children perform least well in both schools. The youngest children in school M perform better than the oldest in school B. However, the mean difference is small. In lesson one the Ambassador and children comment on the high enjoyment levels more than they comment on these in the other two lessons. This is a point worth noting because the mean scores across the lesson show children do less well. The 'novelty' of the visiting specialist is perhaps less important as lessons progress. In addition, children are physically active walking about the classroom instructing the teacher and there was a very relaxed and 'fun' ethos observed in the room by the researcher. However, that does contrast with the data showing children who least enjoy physical activity making the most progress. As the lessons progress, learning reflects more typical programming approaches with children sitting at desks. It is positive that the enjoyment levels are triangulated by the Ambassadors and the children.

Careful consideration is needed with the results because the individual ambassador may influence the class lessons due to personality, communication skills etc. However, the teachers of both classes report that generally the children are motivated to learn and this perhaps mitigates against other influences.

7.4.3 Question level

The forward plans confirm that the level of difficulty and complexity increases with each lesson. Two success criteria with lower scores worth highlighting are those where children predict the behaviour of a given script. Step 2 expects children to understand the tools to apply the CS concepts. In these two examples the quantitative data lower scores is triangulated on by both the Ambassadors and children as being the greatest challenge.

- **Lesson 2** Success Criteria 2.3 mean score 15.7
- **Lesson 3** Success criteria 3.2 mean score 16.1

The lowest score overall was Lesson one SC1.1 I can carry out the steps of a process description as they are given to me. However, in contrast to Lesson 2 SC 2.3 and Lesson 3 SC 3.2 qualitative comments from both final year Ambassadors and children show that enjoyment levels are all high.

Conclusion

The qualitative and quantitative data shows children's progress through an introductory programming course.

The Ambassador's role was critical in planning and implementing the introductory course to children without prior CS knowledge. The study shows that in 3 hours primary school children can progress through the Scottish curriculum achieving standards as expected for their age and stage of development.

The planning approach takes children through the CfE three step hierarchical approach sequentially setting out manageable learning intentions and success criteria for children to be involved in the learning process. It must be noted that the final year Ambassador students have no prior primary school teaching experience, yet children remain motivated throughout the course and succeed. However, the Ambassador's implementation style particularly with classroom management was observed by the researcher to have some limitations and this was supported by qualitative reflections. Both Ambassadors used a direct and didactic teaching style following the lesson script line by line. They were able to

implement the lessons but were limited in asking higher order questioning or modifying their teaching style throughout the lesson to meet the needs of individual children and respond to the challenges.

The Ambassador's strength was in their deep knowledge of CS concepts and the teaching of programming on a multi-level approach which they were able to apply in the new context with younger children. They reflect on this in their journals and how they tried different ways to reinforce the message.

Children's learning experience: Children work in small groups to share ideas and find solutions. All children complete the course covering the CS standard expected in a Scottish Primary school of children up to P4. While age is a factor of progress in each school, it was not a factor across the cohort with the youngest children in one school outperforming the eldest in another school. This was consistent in two out of three lessons. Children's reflections focus more on the activity than their learning, their comments are at surface level. Scratch aims to be intuitive, yet reflections show 50% of children in P3M class do not know 'move right'. There is a need to scaffold further children's reflections of their learning to ensure the learning is embedded. Children continue to achieve at different levels although the margins within the difference are small and at the end of the course children remain motivated and positive attitudes despite the course not covering their expectations.

CS Concept coverage is achieved by all children and over the three lessons there was an increase in the use of children's CS technically appropriate vocabulary. Children enjoy the unplugged version in step 1 but there needs to be a clearer bridge between this approach and how the computational tools use it. However, the most challenge is experienced in step 2 when children predict scripts or follow a process themes. PRIMM (Sentence, Waite, et al., 2017 and much earlier in the use of UMC recognise the importance and challenge of predicting because it requires an understanding of what is being read. Prediction relies heavily on making connections with prior knowledge and is a successful strategy to reinforce learning. It is also recognised in reading comprehension.

7.4.4 Discussion/Future work

The goal of the study is to answer the RQ:

RQ 3: Can a comprehension-first oriented programming course be implemented successfully in a primary school classroom?

The thematic analysis of the Ambassador's and children's reflections on learning provides further insight into the introductory programming learning. The analysis of participant's reflections across the three-lesson show that learning is motivational, and children enjoy the activities. Few children, however, relate the activity to the lesson learning intentions. Prior to the introductory programming course, children show that they have no or limited knowledge of computing science concepts and describe their knowledge in terms of users of technology. A typical misconception by non-specialists including teachers in primary schools. However, as the lessons progress evidence of CS specific vocabulary emerges. Unplugged activities motivate children to learn. Children report positively on the processes and how they learned they, for example, enjoy doing a dance or programming a teacher. However, few make the link between the unplugged activity and using the programming tools, which in this case, is ScratchJr. The key teaching point throughout the course of bridging the real world with the virtual world through understanding the told that use underlying CS concepts is a major focus for the planned learning. This 'bridge' is reported upon by very few children, an issue of understanding that is often challenging for students in introductory programming courses at university level. However, the quantitative data shows that children in the study do manage to modify the games successfully when using the programming tools in Scratch Jr.

Overall, most children progress well and achieve the planned learning covering an appropriate CS curriculum for their age and stage of development within a relatively short timescale of 3 hours. Of importance is the gap between children's expectations of introductory computing science courses and adult's planning a course. Most children want to know about how computers work, how they are built, can be fixed, how apps work, wifi works and how to hack. Enabling children to undertake independent learning lines of inquiry can address this in a manageable way and complement planned learning.

Quantitative data shows a small gap between children who makes the most progress and those who make the least. All children achieve success and gaps in learning are identifiable. Children's reflections on learning change from describing their motivation levels to an increased use of CS vocabulary.

Predictors of success: Overall there is no gender bias towards male and female achievement of success. This is an improvement on the findings in quadrant B where males

achieved greater success than females. However, the two children achieving all measures of success are male. Quantitative data shows a correlation between children's age and attitudes to toys games and activities. There is no relationship between children's attitudes to toys games and activities and their progress through the course. This is positive as the instructional design aims to meet every child's learning needs.

Finally, the introductory lessons, despite using Scratch Jr aimed at very young children worked most effectively for children at P5. Given this introductory course is aimed at the Scottish curriculum first level (P2-P4) this gap needs considered. While subject matter is rigorous, classroom management by the Ambassadors was not surprisingly, at times a barrier for children's learning. In addition, it is unlikely that primary teachers will have this expert knowledge of the Ambassadors to lead the learning.

Ambassadors report high levels of children's engagement and that the planned learning is appropriate. Analysis shows that although children report high levels of motivation, they reflect on the activity and not the intended learning of each lesson. CS concepts covered are in line with expected levels for the age and stage for most of the children. However, there is a mismatch between planned learning in an introductory programming course and children's expectations. Most children report through the 'KWL' (Greenwood, 2019) process that they want to know how a computer works, how it is built and how to fix wifi etc. Very few children commented that they wanted to know how to program. During the study, when describing the programs, most children report at surface level and a few children make the link with the real world. Interestingly overall, all children in school M, including the younger children perform better than the oldest children in school B. This is interesting because there is often an assumption that younger children will progress at a slower pace than older children. However, none of the children had prior CS knowledge and age, in this instance, did not appear to affect progress rates. It is therefore assumed that other aspects impacted on children's progress and this could be their prior experiences. Children with a low interest in physical games are statistically significantly more likely to do better.

Overall, the planned learning aimed to develop children's knowledge of the introductory programming course underlying concepts. The lessons planned to show children the relationship between real life processes and those represented using tools and languages that use CS concepts. Children are motivated by the range of activity based learning and instructional design.

Unplugged activities and use of the Scratch Jr learning environment are reflected upon positively. Scratch Jr was used because the children had no prior experience with Scratch and this was a progressive approach. However, the quantitative data, children's and

Ambassadors' reflections highlight the challenge as children move from real world to virtual worlds. Children appeared to be challenged by the prediction and optimising elements of the lessons. The high levels of motivation are positive, and the planned learning is appropriate for most children. Children make progress covering the intended curriculum. However, the instructional design lacked flexibility for children's personalisation and choice, or facilitating peer interaction. There is a need for further collaboration with primary education learning and teaching strategies to improve further the implementation of the Scottish CS curriculum taking account of the subject knowledge and effective learning and teaching and increased personalisation and choice. The study shows that a comprehension-first approach is successful for primary age children in a classroom setting. It is noted that ambassadors delivered the learning and children progressed; however, it is likely that the skills of a primary teacher would further enhance children's experiences and learning.

Future work

Use of the three-step approach can be motivational and enable children to make progress. Well-planned lessons can be delivered by non-specialists for a short teaching period. There is a need to revisit the Scottish CS curriculum implementation with greater input from primary education specialists and establish if the comprehension-first approach can be applied to typical teaching methods. Delivering any introductory programming course should provide opportunity for children to lead their own lines of inquiry supported by the teacher or educator.

7.4.5 Limitations

This study is an exploration and assessment data is not standardised. The qualitative statements were quantified which is appropriate for this type of study for analysis purposes and clarity. It is important to emphasise that where numbers are used, no inference can be made beyond this sample (Neale, and Miller, 2014).

The Ambassadors have a clear understanding of CS knowledge and use good strategies to reinforce learning. However, the broad banding formative assessment method has limitations. Broad banding of the success criteria resulted has less rigour on what children achieve in each lesson. While their reflections provide some insights there is a need to ensure that additional evidence shows children's progress. This could be through screen

shots of children's work at an individual level. It is not clear each child understands the tiles and scripts which is an effective approach for program comprehension.

Quadrant D: A comprehension-first oriented approach for primary teachers implementing introductory programming in a primary classroom.

Quadrant C reconsiders available materials for teachers delivering introductory programming courses. It shows that a comprehension-oriented approach to introductory programming aligns with the Scottish CfE CS curriculum and impacts on children's progress and their levels of motivation. However, the study was led by the CS expert and does not involve the primary teacher's expert knowledge. Quadrant D is organised in two chapters (chapter 8 and chapter 9). Chapter 8 shows related work around the time of writing in 2018 describing some of the introductory programming approaches in primary school. There remains limited work on comprehension-first approaches in authentic primary school settings and few collaborative studies taking place with primary school teachers. This gap leads to the study in chapter 9 where two primary school class teachers plan and deliver introductory programming courses to their pupils. A comprehension intervention is applied to their approaches and children's progress is evaluated.

RQ2: What typical programming approaches are in place in traditional primary classrooms and how successful are they?

Chapter 8

8.1 Quadrant D:

8.2 Introduction

Findings in Quadrant C show that a comprehension-oriented approach to introductory programming using visual programming languages aligns with the Scottish CfE CS curriculum. The strategy impacts on children's progress and their levels of motivation. However, the study was led by the CS expert and observed by the classroom teacher. Literature on primary school computing science education is increasing to address the difficulties arising when introducing basic programming concepts to young children (Hijón-Neira et al 2020). However, not surprisingly, this increased interest brings a lack of agreement as to the best approach.

The related work section is concerned with teaching introductory programming to primary school children using a comprehension-first approach. Literature shows visual based programming environments continue to be popular and by default promote a create-first approach (Sentence, 2015) with children learning by exploration and at risk of understanding key CS concepts. Gaps in literature show studies in primary school contexts continue to be researcher led with limited input from practising primary school teachers and there is a need to find a way to include the primary teacher's wisdom.

This study in this quadrant uses a researcher-practitioner collaborative approach with a focus on children's outcomes in an introductory programming course implemented in a traditional primary classroom. Two class teachers each plan and deliver a 6-week introductory programming course. Weeks 1-3 they apply their own approaches and weeks 4-6 they apply the comprehension-first oriented approach used in quadrant 3. Findings show that in weeks 1-3 both teachers apply a create-oriented approach using a visual programming environment. In weeks 4-6 they adopt a comprehension-oriented approach which they found manageable and report that it aligns well with how they teach literacy. Data gathered shows that children made good progress using the comprehension-first approach and that they all felt they had been successful. A few children who made limited progress in weeks 1-3 made very good progress in weeks 4-6. Teachers planned the first three lessons and therefore, the intervention was applied in an authentic setting through a balanced researcher-practitioner approach.

8.3 Related work

Literature on primary school computing science education is increasing to address the difficulties arising when introducing basic programming concepts to young children (Hijón-Neira et al 2020). However, not surprisingly, this increased interest brings a lack of agreement as to the best approach. A common starting point in many countries and schools is to teach the Scratch computer programming language to children. Block-based programming environments are motivational, they lend themselves to create-first, an approach that brings quick rewards but some authors consider their use may lead to children not understanding the underlying concepts (Biggs and Collis, 2014) (Lee and Ko, 2015).

Taking account of findings in Quadrant C, this related work section is concerned with teaching introductory programming to primary school children in Scotland using a comprehension-first approach. The comprehension IP approach was first determined by Lopez and Schulte with higher education students and now beginning to emerge in primary contexts. In his work, Lopez (2008), found a strong support for an association between code tracing and code writing skills and an association between code reading and code writing skills of students. Building on Lopez work, Schulte (2008) introduces an educational model of program comprehension. He describes the different aspects of understanding code from the textual (grammar parsing), to the machine/structural (understanding what the code fragments mean at the machine level, to the functional task levels (the connection between code fragments and the parts of the problem that they solved).

Comprehension-first approach

A few introductory programming studies are set within primary classroom settings using a comprehension-first framework. Authors Hijon-Neira, et all (2020) explore learning gains for introducing basic programming concepts in a multi-stage study of 144 school children age 9-12 years. The study focuses on the what and how to teach programming in primary school using Scratch. Researchers employ the use of metaphors and visualisation to introduce basic programming concepts. Although a comprehension-first approach is not overtly stated, the instructional design involves children experiencing scripts and their

function. Researchers (computer scientists) developed and taught the instructional activities with primary teachers contributing by adjusting questions. Children's understanding of a program, its purpose and content were evaluated although children did not create scripts. The study used a multigroup pre-test and-post-test design, with a control group (the use of a blackboard as an unplugged approach) and two experimental groups (the use of a visual execution environment (VEE) with a mouse and the use of the VEE with Makey Makey). Findings show that children in 4th, 5th and 6th grades improved significantly on the test for all grades with large effects. Interestingly, older students do not learn more than younger students, which was not an expected outcome.

Over the past few years, a comprehension-first practice strategy has been the research focus of Salac (2020) who presented a concrete strategy through "Tipp and See". The metacognitive strategy for primary age children, scaffolds learning using procedural engagement through example code. Title, Instructions, Purpose and Play (TIPP) guides students in previewing different aspects of a new Scratch project before looking at code. As a last step children run the code with very deliberate observations of actions and events that occur. Text structures inspire the second stage of the strategy Sprites, Events and Explore (SEE) and provides a roadmap to find code in the Scratch interface. Her study has shown that students using the TIPP&SEE learning strategy vastly out-performed students who did not. This TIPP and SEE framework describes learning to program as highly dependent on reading comprehension at several stages. The quasi-experimental study on 16 teachers supported by researchers, explores metacognition, self-regulation in learning with reflective and constructive learning processes. Teachers and students were randomly assigned to the TIPP and SEE strategy or the comparison group. Findings show that, with the exception of multilingual students, the TIPP and SEE meta-cognitive strategy supports diverse learners in CS instruction. Interestingly, the findings "squares with findings from math and science education, where open inquiry was less effective than scaffolding inquiry for students with disabilities"

Create-first approach

Teachers report that children "want to get on and code" (Bute and Leahy, 2021) and visual programming environments such as Scratch are rooted in the create-first tradition and arguably fulfil that need. Yet, fulfilling this need may lead to challenges with children becoming "frazzled and stuck" (Butler and Leahy, 2021) as reported in a qualitative study

involving 51 pre-service teachers using a creation-first through exploration, trial and error of visual programming blocks.

The following three studies, describe a create-first approach in a classroom setting and exemplify some issues arising. All three studies use the create-first tradition without considering alternative strategies. Findings from the studies show positive outcomes, with results answering the specified research questions. However, potential issues in the results have arguably been overlooked.

Scratch projects are designed by combining graphical blocks to create actions or events for Sprites. These environments are positive because they have a 'low floor' and students cannot make syntactical mistakes. "Despite the affordances of graphical tools, programming is cognitively complex, and rich conceptual mental models may not emerge spontaneously. (Häkkinen, et al, 2021). The following study in Sweden with classroom learning sessions directed by researchers in the classroom involved 30 pupils age 6-7 working in pairs. The aim of the study is to investigate pupils' multimodal representations when they use block programming. Although findings are positive with pupils having positive attitudes, high self-efficacy and learning is visible, the results of the questionnaire show that 55% of children view programming as difficult although 86.2% would like to have more programming at school.

Each study in primary school introductory programming poses different research goals with a number interested in gender. One such study (Charoula and Valanides 2020) focusing on gender issues in programming involves 50 children age 5 to 6 across eight urban schools. The instructional design shows children learn programming through problem solving scenarios. For example, in one scaffolded approach, children use laminated command cards to build a sequence for each task to direct a floor robot. In another approach, children collaborate with the researcher using think aloud strategies. Authors' hypothesise that the use of robotic activities with the use of small programmable floor robots is an effective way for developing young children's programming skills. This study does not overtly promote or comment on its creation-first approach and while it is positive that the approach did find insignificant gender differences, there remains a wide variation in assessment scores more broadly indicating a gap in progress across participants irrespective of gender.

The primary school introductory programming classroom

Primary education is a complex phenomenon mainly because primary teachers plan using intuition and experience. Primary teacher lesson planning takes account of both the

learning environment and the instructional design. Researchers recognise the gap in understanding fully the complexities between educational research and practise (Biesta, 2007). Given the newness of the computing science and introductory programming in formal primary school education, studies in natural settings to date are limited, research is time consuming and there's no guarantee that the interventions will work. However, engaging in research through reading, reflection and or professional dialogue can provide teachers with an alternative view that they can evaluate and consider.

Researcher-practitioner collaborations

From literature there are three types of researcher-practitioner collaborative studies: Teacher focused, student focused and teacher and student focused. When compared with the impact of collaborative professional development, studies of individually oriented professional development offer only weak evidence of its capacity to influence teacher or pupil change (Cordingly et al, 2005). Teachers learning effectively with and from other professionals is reinforced in the literature as a powerful component of effective professional learning. Purposeful collaboration between peers is also a feature the world's greatest school systems (Mourshed, Chijioke and Barber, 2011). Teachers value learning with other teachers (Day et al, 2007) and many teachers involved in focused collaborative professional development subsequently change or substantially develop aspects of their teaching which improves their pupils' learning. However, scholars recognise issues across many disciplines including computer science, practitioners want timely solutions to impact positively on the children they teach, whereas researchers dedicate themselves to probing theories over a long period. In addition, collaborations may suffer because of the barriers in communicating across different knowledge bases or a power imbalance.

The introduction of CS in schools is supported by several initiatives and programming environments such as Scratch or Kodu; curriculum resources; guidelines; lesson plans and materials. In addition, as discussed in quadrant B of this thesis, researcher-led studies typically gain evidence-informed insights into introductory programming. Certainly, the research informed insights on introductory programming in primary schools can be both useful and purposeful and helps teachers expand their mindset and reflect on practice. However, for sustainability and replicability researchers may need to consider, "If I am a hard-pressed teacher, what is most likely to encourage, support and sustain me in

developing and changing my practice to meet the needs of the students. What is likely to engage my attention rather than feel like another irrelevant dictat” (Webster et al, 2012).

Collaborative learning is most likely to be effective where attention is paid to developing trust, building on existing relationships and networks, recognising respective roles and contributions, ensuring knowledge meets local needs and addressing competing priorities. (Sebba, Kent and Tregenza, 2012). Researcher-led approaches typically used in primary school introductory programming courses, may have an imbalance in the level of influence. In addition, the establishment of equity in collaborative relationships may be constrained due to research practices, culture and hierarchies. No one form of collaborative learning outshines, but powerful modes of collaborative learning include lesson study, learning walks, instructional rounds, and coaching and mentoring (Cordingley et al, 2005). Collaborative working, being observed, receiving, and providing feedback have greater commitment to changing practice when trying out new strategies. At its heart this involves mutual engagement where colleagues open up, share and co-construct ways of developing practice (Fielding, et al, 2005).

Primary classroom learning design.

“there is considerable evidence from different studies suggesting that how teachers behave in the classroom, the instructional approaches they employ, significantly affect the degree to which students learn” (Van Tassel- Baska, Quek, and Feng, 2007).”

Some work has been undertaken to explore and understand instructional designs of a primary school teacher in a natural setting (Fatourou, et al,2021). A review of primary school learning design choices of primary school programming courses in empirical researchers aims to frame learning situations and methods to overcome the lack of extensive experience and designing a programming course. Fatourou’s review recognises the limitations of conclusive field proven teaching practises to use as a basis for instruction design an introductory programming. To overcome the lack of extensive experience in designing a programming course, their study reviews learning design choices of primary school programming courses and empirical research. Many studies who are excluded because learning procedure was not available. Findings from the 22 studies in Fatourou’s review show the most common language used scratch. Teaching strategies such as PBL and games and CSS unplugged activities to introduce ideas that students will encounter later. Organisation of instruction and classroom layout shows programming in peers in groups. Authors (Fatourou, 2021) conclude that efficient curriculum structure, class

teaching and learning methodologies differ. Learning situations are usually described by many variables. Moreover the methods employed are sometimes impossible to follow in specific classroom context. Of note is that authors make no mention of the creation first or comprehension-first approach nor do they highlight if studies are practitioner or researcher led.

Conclusion

Introductory programming is an emerging new subject in primary school curricula. Teachers will plan learning for children taking account a range of variables including their own experience and available guidance. Visual based programming environments continue to be popular and by default promote a creation-first approach with children learning by exploration and at risk of understanding key CS concepts. Gaps in literature show studies in primary school contexts continue to be researcher led with limited input from practising primary school teachers.

Chapter 9

Introductory programming studies in traditional primary school classrooms.

9.1 Introduction

The recent introduction of CS education in primary schools, not surprisingly, brings limited pedagogical knowledge. Children at a younger age experience introductory computing science in formal education. Introductory visual programming languages such as Scratch are often delivered, in the main, by non-computer science specialist primary teachers to meet the curriculum standards. However, emerging research shows some young children face difficulty when first engaging with CS.

The studies in this final quadrant explore primary teacher's planning and implementation of a CS course in two schools. This case study explores children's socioeconomic status and introductory programming approaches delivered by non-CS specialist primary teachers in a formal education setting. Each study continues with the iterative nature of the overall thesis study design, the studies and their findings are and is presented separately.

The first study in School B is followed by the second study in school K.

9.2 Method

9.2.1 Aims and Objectives

The goal of these studies in school B and school K is an exploration of primary teachers approach to computer science education and its impact on children's progress in both covering and understanding CS concepts. The use of comprehension-first approach is considered.

RQ2: What typical programming approaches are in place in traditional primary classrooms and how successful are they?

RQ3: Can a comprehension-first oriented programming course be implemented successfully in a primary school classroom?

9.2.2 Overarching Approach

This quadrant consists of two studies (Study A and Study B) undertaken independent of each other in two separate primary schools. The research for both studies is broadly similar

with a focus on the Teacher's role, children's learning process, predictive variables and CS concepts (Fig. 9). Study B includes some additional predictive variables measures for analysis.

9.2.3 Research Design for both studies

Both studies involve of 6 x 1 hour introductory programming organised into two distinct phases. In phase one teachers plan and implement a 3 x one-hour introductory programming course. The second phase of the study applies a 3 x one-hour intervention to the general teaching approaches undertaken by the classroom teacher. The intervention is then implemented to the same cohort of children. The intervention takes account of CSED research and the Scottish curriculum three step comprehension-first oriented approach. The study retains the overarching framework for consideration (Fig 10).

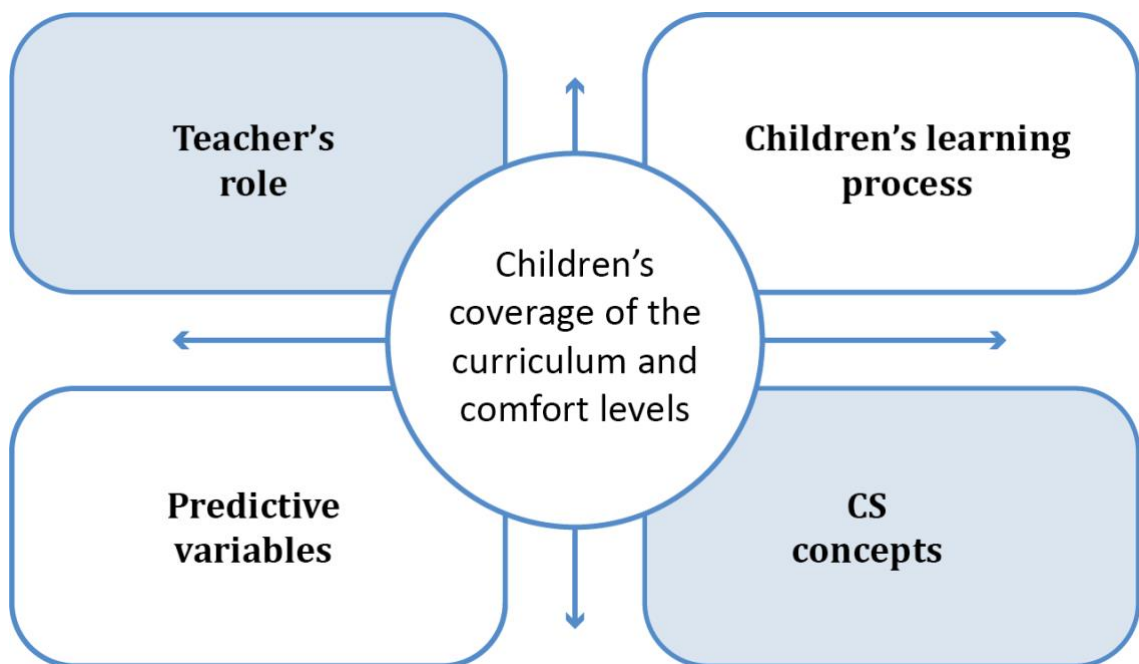


Figure 9 The study evaluation framework

9.3 Study A – School B

9.3.1 School B – Setting

The study takes place in a typical primary school setting. Children undertake the lessons in their usual classroom environment.

9.3.2 Participants

School B is an inner-city school with broad ranging SIMD catchment or SES status.

School B participants attend a large inner city primary school with a diverse SES demographic. 53 children aged 8-9 living in a range of socioeconomic areas participate in six hours of instruction delivered by their class teachers over six lessons.

The school is at the early stages of implementing a new national computing science curricula. The sample from the school includes two class teachers with more than five years teaching experience each and no prior experience in teaching computing science. Prior to the study, none of the children had experience of programming in a formal school setting. For the study, 54 children age range eight to nine, are organised in two classes.

Class one (n=24) Class two (n=29) with a range of SES backgrounds. All participants, including the children's parents, receive a plain language statements outlining the purpose of the study, expectations of their involvement and dissemination of findings. Of those invited to participate, 100 percent consented. The plain language statements and consent forms have ethical approval.

Participants' socioeconomic status ranked by SIMD deciles⁵. Geographical areas (called data zones) are categorised from the most deprived (ranked 1) to least deprived (ranked 6,976)⁶

SIMD rank by Decile	Data zone rank		No. of children in class living in each data zone	
	From	To	Class 1	Class 2
01	1	697	1	1
02	698	1395	2	1
03	1396	2092	4	2
04	2093	2790	3	3
05	2791	3488	5	4
06	3489	4185	0	4
07	4186	4883	0	3
08	4884	5580	4	7
09	5581	6278	4	0
10	6628	6976	1	1
No data			0	3
Total			24	29

Table 50 Children's SES status measured by SIMD deciles and data zones

Children's socioeconomic status (SES) measure is a combination of factors including education, income and occupation. There is evidence of the negative impact of poverty on children's academic progress more broadly (Boorks-Gunn, and Duncan, G., 1997) (Sirin, 2005) and STEM specifically (Hoffer et al. 1995). The scale used in the study is the Scottish Government Social Index of Multiple Deprivation. The children living in areas

⁵ <https://www.gov.scot/publications/simd-rank-to-quintile-decile-and-vigintile/>

⁶ [Scottish Index of Multiple Deprivation 2020 - gov.scot \(www.gov.scot\)](https://www.gov.scot/collections/scottish-index-of-multiple-deprivation-2020/?utm_source=redirect&utm_medium=shorturl&utm_campaign=simd) available at https://www.gov.scot/collections/scottish-index-of-multiple-deprivation-2020/?utm_source=redirect&utm_medium=shorturl&utm_campaign=simd accessed 02/01/2022

with the highest levels of deprivation on a scale 1-10 are recorded as '1'. Children with the least levels of deprivation are recorded as '10'.

9.3.3 **Materials**

The intervention places a high priority on children talking to each other and the teacher about the processes in the Scratch animations. The study aims to improve their understanding of the Scratch blocks, structure and functions.

9.3.4 **Procedure**

The objective of this exploratory case study is to understand how computing science curriculum delivered by non-specialist primary school teachers impacts on children's progress. It explores the use of a comprehension-first oriented approach within a primary teacher's typical classroom environment. The intervention takes account of what the teachers planned and implemented in phase one. On screen application of computer science concepts in Scratch projects are evaluated

Phase one: Teacher's plan their lessons independently of the researcher. The researcher observes the lesson and notes each section for replication in the intervention. The primary teacher's lesson structure forms the basis of the intervention and can be viewed below.

Phase two of the study applies the comprehension-first oriented approach to primary teachers approach that they implement in phase one. It consists of a weekly meeting with class teachers and the researcher. Teachers deliver one hour weekly sessions for a period of three weeks. Materials for phase two were prepared by the authors. The author in consultation with the class teacher creates materials based on those implemented in phase one. Weekly meetings with teachers involve discussing the prepared materials to support implementation and explaining the thinking behind the methodology. Authors place an emphasis on children observing and talking in advance of building the scripts. They use a modified version of comprehension-oriented approach. Teachers make suggestions for delivery based on assessment for learning strategies (Gardener, 2012). Typical classroom management strategies are also sustained in phase two. Children in the classes are familiar with particular classroom routines including those that encourage discussion.

Phase two lesson one. Part one 5 minutes; children recap on what they have learned and complete the class chart that they started in phase one.

Each teacher shows working animations created by the researcher from the phase one contexts "Chase the Bug", "Disappearing Wizard" and "Busy Street" that they had planned. The children have not seen these working fully or presented in this way before. In trios, children discuss what is happening in each of the animations and report back to the class. The class build an oral 'story' around what they are seeing including backgrounds and sprite actions. Each trio is a detective trying to find something new. The class teacher places a strong emphasis on teamwork. The class vote on an animation to explore and after revisiting their description the blocks for each script are shown. This helps the children put meaning to the blocks in context. Again, this is repeated and children suggest script modifications. They are encouraged to be creative in their modifications for the animations.

Children work individually at a computer with script prompts for each animation. They choose an animation from the demonstration which have the scripts completed. They perform simple modifications. During this time, the teacher and researcher speak to the children in trios at their computer and ask them to talk about their projects and describe their scripts. Children record their understanding of the scripts line by line on their prompt sheet.

Children create exit passes describing what they learned and traffic light their progress.

Phase two lesson two and lesson three. Part one 5 minutes: children recap on what they have learned and complete the class chart. Part two 10 minutes: A few children (creators) present their projects from the previous week and lead discussions with their peers. The creators then show their scripts to the class and describe the function of each block. Children are shown how to open projects in Scratch and access prepared animations that they can modify. Part three 30 minutes: Children work individually at the computers modifying and making additions to prepared animations from the class shared folders. They are encouraged to create their own games. During this time the teacher and researcher speak to the children in trios at their computer and ask them to describe their scripts. Children's progress and comments are recorded.

Part four 15 minutes: Plenary session: Using no hands up strategy and a few children's projects are selected. Children lead discussions with their peers on what is happening in the

animation. The creator shows their scripts and describes each block's action. All children complete their exit passes outlining what they have learned.

9.3.5 Data analysis

In both studies children's SES status is considered.

9.3.6 Interrater reliability

Construct validity: a clear protocol for the evaluation rubric was created moderated by two PhD students from the University of Glasgow CCSE. They reviewed the teacher's planned activities and with the author created a rubric to evaluate children's progress in phase one and in phase two. After reviewing the materials independently, they compared notes and amend the rubric. The same two PhD students with expertise in statistical modelling supported and moderated the statistical analysis for this study. They had no other involvement in the study. Evaluation of individual screen shots was determined in collaboration with the class teacher.

Data sources in the study are as follows:

- (1) Children's age
- (2) SES
- (3) Progress in phase one
- (4) Progress in phase two.

The analysis considers the dependent variable of progress in the two phases of the study. The overview of themes organises the qualitative and quantitative measures for the results. Firstly, the role of the teacher and how they lead and manage the learning is observed. Next the role of the children and their cognition as a result of the teaching methodology. In addition, the CS concepts covered within the introductory programming course need considered as there are national expectations of CS experiences. The final consideration are variables that may impact on a child's progress such as age, gender, SES, reading and

numeracy attainment and their interests in toys, games and activities which may contribute to their success in the introductory programming course.

The quantitative evaluation of children's socioeconomic status and their progress measured in the following three dimensions:

Dimension one: Scratch Environment Tools

Dimension two: Computing science concepts linked to the national curriculum

A qualitative evaluation of children's SES and their progress is quantified, these include:

Teacher's prediction of success

Children's evaluation of their own progress

Dimension three: Code comprehension using pen and paper task

Due to the study involving children, no control group and the intervention following phase one, it was critical that the analysis was independent, rigorous and robust before undertaking further study.

9.4 Study A School B Results

The following studies in two schools (School B and School K) with contrasting demographics take account of quadrant C recommendations. School B is an inner-city school with a diverse catchment area measured by the Scottish index of multiple deprivation (SIMD). School M is a rural school with children living in an area of high measures of deprivation.

The study in school B applies a comprehension-first oriented approach to introductory programming planned and implemented in a traditional classroom setting by primary school teachers. 53 children age 8-9 participate in a 6 hour course organised into two parts. In part A, the teacher plans and implements 3 x 1 hour session using a create-first approach. In part B, the teacher applies a comprehension-first oriented approach to their CS planning while maintaining their typical classroom design. Findings for school K take account of the progress made by each child across the 6 hours. The foundations of socioeconomic inequities and the educational outcomes of efforts to reduce gaps in socioeconomic status are of great interest to researchers around the world, and narrowing the achievement gap is a common goal for most education systems. Therefore, the study also considers children's SES status. Findings show that children make better progress in phase 2 and have higher levels of motivation.

School K case study also includes a 6 hour course and broadly follows the same design as school B. It looks at findings in more details considering (i) the teacher's role, (ii) the child's role, (iii) CS concepts and (iv) additional variables of gender, age, SES, reading attainment, literacy attainment impacting on children's progress and attitudes. In phase 1, teachers plan and implement a 3 x 1 hour course. In phase 2, teachers plan and implement a 3 x 1hr intervention taking account of the Scottish CS curriculum the 3- step comprehension-first approach. In phase 2, primary teachers maintain their typical classroom design and management from phase 1. Findings show that in phase 1, teacher's select lessons from code.org, children work individually self-learning covering less than one third of the curriculum. Gender and reading attainment impact on their progress and view of level of difficulty. In phase two when the intervention is applied, they provide small direct teaching inputs facilitating discussions, children work in pairs looking at the scripts on different levels before completing online activities independently. As a result, there is greater coverage of the curriculum. In phase 2, there is an association between gender and progress. Although all children do make much better progress and their view of the level of difficulty improves.

Following the moderation activity, one change was made to the rubric and included technical vocabulary for greater accuracy. It is noted that Scratch requires two distinct skills: firstly, the use of the tools to create backgrounds etc and secondly computing science concepts such as variables, loops and sequence. The table below shows the coverage of these skills in the two phases and scoring. Where a task required a particular skill the following scoring system was applied. Each script is measured using a 0-5 scoring against the rubric as follows:

- 5** – Complete achievement of the goal, task or understanding
- 4** – Mostly complete achievement of the goal, task or understanding
- 3** – Partially complete achievement goals
- 2** – Very incomplete
- 1** – Did not complete any part of the goal task or understanding
- 0** – Made no attempt to complete or did not reach this part of the task.

The rubric focus was on the computing science concepts. The data gathered for the use of tools was not included in the analysis.

Animation	Phase 1			Phase 2		
	1	2	3	1	2	3
Environment tools/set up						
Backdrop	∅	∅	∅	∅	∅	∅
Sprites	∅	∅	∅	∅	∅	∅
Resize	∅	∅	∅	∅	∅	∅
Open existing projects				∅	∅	∅
Run animations accurately				∅	∅	∅
Copy from crib sheets				∅	∅	∅
Modify scripts				∅	∅	∅
Assign multiple actions to multiple sprites	∅	∅	∅	∅	∅	∅
Assign multiple actions to multiple sprites	∅	∅	∅	∅	∅	∅

Table 51 Learning outcomes

Table 52 shows the learning outcomes assessed in the study. ∅ denotes learning that is assessed and a score awarded.

9.4.1 Results

9.4.2 Planned learning

Phase one The two class teachers plan three lessons collaboratively taking account of the National curriculum guidance. They select experiences and outcomes alongside assessment measures relating to sequence, variables and loops. Both teachers choose a visually attractive commercially produced series of lessons with predetermined learning intentions and success criteria. Prior to implementation of phase one, the teachers invited the local secondary school computing science teacher to deliver an introductory didactic Scratch session to the children. During this session, children watched the secondary teacher go through the interface. Children did not get an opportunity for hands on exploration of Scratch on the computer because of time restraints. This introductory lesson is not included in the evaluation of children’s progress. However, two children from the introductory session are identified by the class teacher as future ‘technical support’ for the six sessions. Although the two children had no prior experience of Scratch they had a very high interest in the use of computers. Throughout the study they provide basic technical

support to their peers and the class teacher. They set up the computers prior to each session, assisted with log in information and generally made themselves available. They were known as the class ‘digital leaders’.

Each class lesson follows the following instructional design:

Part one: 15 minutes whole class lesson. Children complete a class chart on what they know about computers and programming.

Part two. 30 minutes, children work individually on screen completing the commercially produced task sheets (Twinkle, 2023). They self-select from three ‘chilli challenge activities’. Chilli challenges are presented by the teachers as incremental in difficulty.

- 1) Catch the Bugs - one chilli
- 2) Disappearing Wizard - two chillies and
- 3) Busy Street - three chillies.

The publishers differentiate the content of each of the tasks by level of difficulty. An analysis of the tasks shows that they have broadly similar computing science concepts.

Part three. 15 minutes plenary session. A few children share with the class what they achieved.

9.4.3 Results in progress

The first hypothesis is that the employing a comprehension-first approach will improve children’s performance in phase 1 and phase 2. It had been predicted that phase 2 would show an improvement. Upon examining the distributions, Phase 1 was found to have a strong positive skew, with 14 children obtaining zero points. This suggests a floor effect, where the task difficulty level was too high (relative to prior knowledge) to reliably distinguish among high-performers and low-performers. This floor effect vanished in phase 2. From the summary statistics tabulated in the table, as well as the boxplots in Figure 10, there is a difference in central tendency of a large effect size. A Wilcoxon signed rank test was conducted, revealing a significant difference ($V=49$, $p<0.05$), leading us to reject the null hypothesis relating to a performance difference in phase 1 and phase 2. Data suggest that a relationship exists between the comprehension-first approach and children’s progress.

	Mean	Median	SD	Skew
SIMD	11.14	10	4.7	-0.17
Phase 1	9.75	8	9.72	1.14
Phase 2	21.75	20	10.62	0.56

Table 52 Summary of statistics

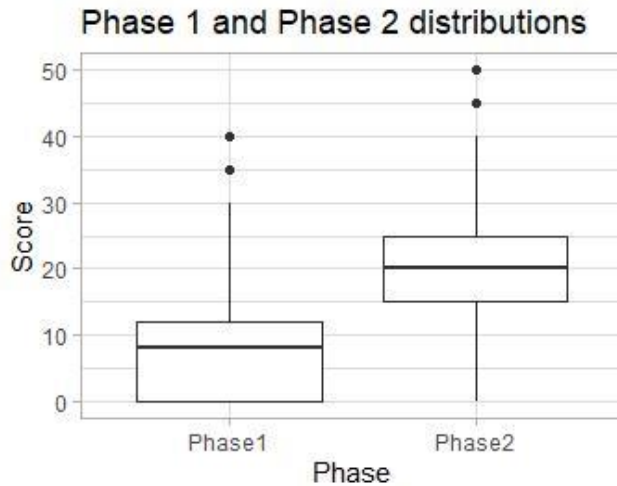


Figure 10 Boxplot comparing phase 1 and phase 2 distribution

The second hypothesis stated that the relationship between scores and socioeconomic status would be attenuated by the phase 2 training - in other words, that the regression slope would be flatter in phase 2. The scatterplots for phase 1 and 2, along with trend lines, are shown in Figure 11. No linear relationship between performance and socio-economic status is evident in either phase, and the regression estimates for SIMD on phase 1 (estimate=0.095, T=0.31, p=0.76) and phase 2 (estimate=0.08, T=0.26, p=0.8) are both non-significant. This concludes that the null hypothesis cannot be rejected.

The middle lines indicate medians, while the hinges indicate quartiles.

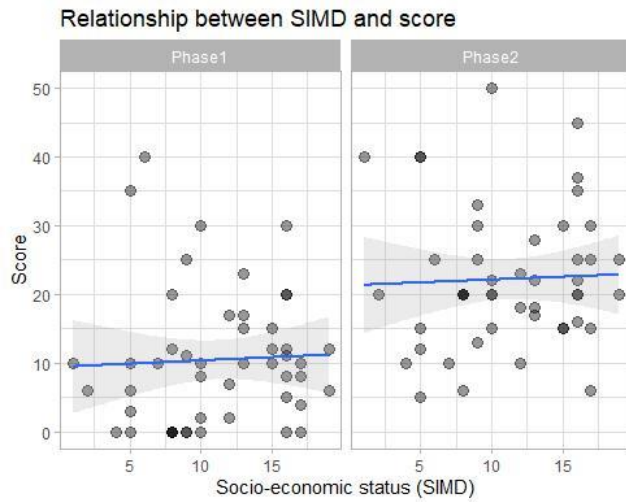


Figure 11 Scatterplots of SIMD and phase 1 and phase 2 scores

9.4.4 Qualitative data

In phase one a few children obtained a score of zero and their qualitative comments include:

Child B1b: "I'm bored"

Child B1v: "I didn't learn much"

Child B1w: "I couldn't get it to work"

By phase two comments were almost all positive

Child B1b: "I am so happy doing this. Look, look I got it to work"

Child B1v: "I now know what's going on behind the animation and how it works"

Child B2ac: "I know what a loop is and I wanted to add arrow keys because noone else had done that"

Teachers involved in the study felt valued and that their understanding of effective learning and teaching in a traditional primary classroom setting had informed the intervention. They stated that the phase two approach was meaningful to them: "It made sense".

9.4.5 Discussion/Future work

The intervention is set within a comprehension-first oriented approach. The critical element of the intervention is for younger children, the modified approach used in the intervention employs a focus on listening and talking about the animations before reading or writing the scripts.

The emphasis at the outset on talking about the bigger picture of the animations before moving to the detailed orientation of individual blocks was positive for children in the study. This minor adjustment to classroom practice was applied by the classroom teacher to commercially produced materials. It provides children with focused opportunities to improve their vocabulary and put meaning to the text structure enabling understanding of its function.

Program comprehension focuses on reading comprehension before writing. The intervention takes this approach one step further by focusing on listening and talking before reading. Listening and talking before reading and writing in literacy is well-understood by primary teachers. The use of listening and talking has informed the teaching of reading to young children and early literacy development for decades. As babies, children have months of listening before they form words which they can apply to reading text and then writing. It is proposed that the process is similar for learning a new block-based programming language. However, program and text comprehension is not in scope of this study but remains a consideration for planning interventions for CS education in a younger setting.

Results show that by the end of the 6 hour teaching block; all children made progress irrespective of their SES status. Across the board there is a balanced playing field for SES. Phase two shows overall progress accelerates by comparison to phase one and children's enjoyment levels increase. Arguably in phase two, the children became more familiar with the Scratch programming tools. However, the assessment of the projects show more children across the sample apply more CS concepts accurately. Qualitative data shows a few children articulating a deeper understanding of the Scratch blocks. Given the significance in data of improvement, there is confidence that this progress is not attributed solely to children experiencing more time on the visual programming environment Scratch. While this is positive, there is an issue that still needs addressed. There remains a normal distribution curve for all children and more differentiated learning is needed to disrupt possible patterns arising over a longer period of time.

If the study is upscaled then an automated assessment tool should be used to verify and complement manual evaluations as they could be subjective.

Future work:

With the limited primary CS education literature available, this study contributes well to the growing interest and knowledge base on comprehension-oriented approaches in primary school contexts.

The study concludes that young children benefit from facilitated discussions about the processes involved in each animation and understanding the overall goal of the script. This big picture thinking before looking at the detail of the visual programming tool blocks deepens their understanding of text structure enabling more effective program execution.

While more work needs to be done, there is confidence that this approach disrupted typical SES patterns of underachievement in education performance with all children achieving success. It is proposed that equitable collaborations with effective non-specialist practitioners in primary school education and CS experts must continue. Building knowledge through joint working has the potential to advance understanding, maintain the integrity of each discipline and improve outcomes for young children.

9.4.6 Threats to Validity/Limitations

Internal validity: Although there is no control group this is an exploratory case study to inform future work. Confidence is needed that the intervention would not disadvantage any child in any way. Given the approach impacted positively on children over the six week period, there is confidence that this study could be repeated with phase two approach first without detriment to the children. This should be considered for future work.

9.5 Study B – School K

Introduction

School K study has the same aim and overall goal as School A. However, the iterative nature of the thesis lends itself to some additions to the data collection and analysis. The second study in school K takes account of existing knowledge of CS introductory programming courses (independent variables) such as age, gender, SES, reading and numeracy attainment. In addition, it explores possible predictors of success through considering the relationship between children's acquisition of STEM skills and knowledge acquired outwith formal education through play.

9.5.1 Aims and objectives

RQ2: What instructional designs do primary teachers adopt in introductory programming?

RQ3: Can a comprehension-first oriented introductory programming course be implemented successfully in a primary school classroom?

9.5.2 Approach

In the second study (school K), observations of the teacher's role, the children's learning process, CS concepts covered impacting on children's progress are explored. In addition, relationships between predictor variables (independent variables) such as gender, SES status, reading and numeracy attainment and play interests are correlated with children's progress through the course and their view of the level of difficulty (comfort levels) of the course. In the second study data collected explores the role of teachers before and during the intervention

Predictors of success (Independent Variable)

CSED research suggests a relationship between gender, socioeconomic status, reading and numeracy attainment with success in introductory programming. The average successful completion rate in IP at HEI has been 67%. (Watson and LI, 2014) The percentage of computer science bachelor degrees awarded to females has declined from 37% in 1986 to 18% between 2005/2006 and 2012/2013 (National Centre for Education Statistics, 2014). In addition, there is a strong positive association between SES and academic achievement. Wilson and Shrock (2001) and Silver (1982) found some correlation between math and verbal SAT scores. With CS as an entitlement for younger children, there is a need to ensure approaches at an earlier stage in education do not perpetuate these issues. Research shows a relationship with children's experience and interests in toys, games and activities outwith school such as block play and construction toys and achievements in STEM subjects. Therefore, the study lends itself to provide insight into each of these variables.

There is evidence that boys obtain greater experience with computers at home than girls (Comber et al., 1997). This experience brings familiarity in using computer hardware and perhaps explains transferable skills. There is a link between experience in using computers

and positive computer attitudes. Brosnan, (1998) suggests that boys are more interested in computers than girls. Gender is therefore worth exploring given girls and boys are accessing more digital technologies in their everyday life and it could impact on children's progress more longer term. The study analyses gender and its relationship with children's progress during introductory programming in both phase one and the intervention phase two. Females in the study are coded as '0' with males coded as '1'.

9.5.3 Study B School K Research Design

School K is a rural school with children living in area of high deprivation

The second study in Quadrant D takes place in a rural setting with children living in areas of high deprivation. It follows a similar design as the first study and additional exploration of predicting success.

Measures include:

- The role of the teacher
- The children's learning process
- CS concepts
- with progress and children's view of the level of difficulty.
- Dependent variables: Progress in phase one and phase two

This study is designed to explore the research questions above through the delivery of introductory programming course in a primary school. The report outlines how primary school teachers plan and deliver computing science education (CSED) in a 6 x 1 hour introductory programming course. The experiment is organized into two phases, phase 1 the teachers plan 3 x 1 hour lessons and deliver their own material. Phase 2, the researcher and primary teachers plan collaboratively, and the teacher's deliver a further 3 x 1 hour lessons.

9.5.4 Participants

From the sample, 3 participants met the following selection criteria: Qualified primary school teachers with 5 years teaching experience, no formal CS teaching qualification, teaching P6 children and within reasonable travelling distance for the author. The participant chosen for this study is male and works in a rural school in an area of high deprivation. The socioeconomic status and rurality provide further diversity with the sample of schools used in the thesis.

After contacting the teacher, permission was sought from their headteacher and local authority for the study to take place. Another teacher volunteered to participate, giving two classes from the school.

Table 54 shows the total number of children participating by class, their age, the resource used by the teacher to implement the course and the corresponding Curriculum for Excellence computing science level. Children’s SES status is measured by the Scottish Index of Multiple Deprivation (SIMD). SIMD is a relative measure of deprivation across 6,976 small areas (called data zones). If an area is identified as ‘deprived’, this can relate to people having a low income but it can also mean fewer resources or opportunities. SIMD looks at the extent to which an area is deprived across seven domains: income, employment, education, health, access to services, crime and housing.

Class	Code	Teacher		No.	Chn		Age		SES (SIMD)	Resource	Course
		F	M		F	M	Mean	Mean			
P5	A	0	1	26	16	10	10.2	1.3		Code.org	C-D
P6	B	0	1	28	12	16	9.3	1.2		Code.org	D-E
*Curriculum for Excellence computing science level planned by teachers.											

Table 53: School B participants:

The study has Class A n=28 pupils and Class B n=26 pupils. The two teachers participating are male.

The data presented differs from school B because of improvements to the study and data collection approaches.

Table 55 shows children’s grouping in each class by their CfE reading attainment levels. Children’s reading attainment and numeracy range from 1.1 “well-below average”; 1.2 and 1.3 “below average”; 2.1 “average” 2.2 “above average”. Both teachers have over five years teaching experience, one teacher is familiar with using code.org as a resource for teaching CS to primary school children which was the reason this resource was chosen. Neither teacher have a background in CS.

Class	CfE Reading					CfE Numeracy					Total
	1.1	1.2	1.3	2.1	2.2	1.1	1.2	1.3	2.1	2.2	
A(P5)	5	2	1	6	12	4	2	4	4	12	26
B(P6)	7	1	6	7	7	6	4	9	9	0	28

Table 54: Children grouped by reading and numeracy CfE levels.

9.5.5 Materials

The traffic light system is a process that the children are familiar with across all areas of their learning. For the purpose of the study, this approach is used by the children to report on the level of difficulty for phase one and phase two.

9.5.6 Procedure

School K study reflects school B. There is a 6 x 1 hour introductory programming course is split into two phases. Phase one, teachers plan and implement 3 x 1 hour of lessons. In phase two, the teachers plan alongside the researcher to amend teaching materials and implement the intervention taking account of a comprehension-first approach.

CS content measurement: Children’s progress throughout the introductory programming course is measured using a 5-point scale in line with the studies throughout the thesis. 5 points are awarded for completing a lesson or series of lessons covering a particular CS activity. The unplugged activities were omitted by the teachers and therefore not included in the scoring. If a child starts Course C and completes lesson 2 sequencing activities, they receive a score of 5. If they complete Lesson 2 activities and Lesson 3 debugging loops activities, they receive 10 points. There is an expectation that children go through each lesson as prescribed by the course. By the end of course C they can accumulate a total of 25 points. Children retain their score of 25 as they move to Course D where they accumulate 5 points for each of the lessons in Course D. The maximum score for completing Course C and Course D is 65. It should be noted that on occasions, more than one lesson covers a CS concept. For example, lessons 5 and 6 cover loops and a total score of 5 is applied for completion of these lessons. In addition, Course C lesson 5,6 loops is

less challenging than lessons 8,9 and 10 loops yet the accumulative lessons receive 5 points.

9.5.7 Data analysis

The role of the teacher in each phase is gathered through observations; the children's learning processes are gathered through observations; the CS concepts covered data collected through observations of planning materials and converted to quantitative measures; the prediction variables are quantitative measures. The following section details the measurements and scoring for each of the four themes: The role of the teacher; the role of the children; the CS concepts and the additional variables.

The computing science concepts in the study are gathered from the resources used by the teacher and recorded in their planning formats. The CS concepts used are synthesised using the Curriculum for Excellence CS experiences and outcomes and benchmarks.

Coverage of the computing science content and children's comfort level in the introductory programming course are used to measure the impact of the teacher's role, the children's learning process and the predictive variables correlation.

Measurements/scoring

This section outlines the measurements and scoring used for each of the four themes that impact on the data analyses in the results section.

The role of the teacher

The teacher's actions during phase one and phase two are recorded using written field notes under the headings; content; pedagogy; student development; instructional strategies and planning. The notes were synthesised using the General Teaching Council Standards Teaching and Learning: Classroom Organisation and Management. These standards relate to professional actions expected of a registered teacher in a classroom setting.

The children's learning process

Field notes record the children's learning experiences and the learning (cognitive) processes in which they are involved. These notes are synthesised using Curriculum for Excellence computing science experiences and outcomes; cognitive processes for Schulte's block model for program comprehension and active learning approaches.

Children’s view of the level of difficulty throughout phase one and phase two of the study were captured using the traffic light system. Children are familiar with the traffic system in other areas of their learning. They use the colour red to indicate that they found the work tricky and don’t understand it; yellow is used to indicate that they found the work ok and that they understand it although they could not explain it to someone else and they used green to indicate that they found it easy and that they could explain it to a friend. Each response has a score ranging from 1 point for red, 2 points for yellow and 3 points for green. Table 56 provides an overview of the statements used by children to describe their view of the course level of difficulty.

SCORING FOR EACH STATEMENT

Colour	Statement	Score
	“I found this hard or tricky, I don’t understand it yet”	1
	I think that I could understand it, but I could not explain it to someone else”	2
	“I found this easy and I could explain it to a friend”	3

Table 55 Overview of the traffic light self-assessment statement scoring and children's view of the level of difficulty

The CS concepts

Course coverage

Table 57 shows the composition of each CS activity and concept within Code.org Fundamentals (2017) Course C, Course D and Course E. The lesson, course and scoring measure (SCO) are included to provide an overview of the lesson scoring. Where there is an unplugged lesson no points are awarded because these were not used by the class teacher. Each CS concept has at least one or more lessons and the score awarded for each CS concept is 5.

LESSON	COURSE	SCO	LESSON	COURSE	SCO	LESSON	COURSE	SCO
	C		N	D		N	E	
1	Unplugged	0	1	Unplugged	0	1	Unplugged	0
2	Sequencing	5	2	Sequence	5	2	Sequence	5
3	Debugging – loops	5	3	Events	5	3	Unplugged	0
4	Unplugged	0	4	Nested	5	4	Debugging	5
5	Loops	5	5	Loops		5	Loops	5
6			6			Unplugged	0	
7	Unplugged	0	7	Unplugged	0	7	Loops	5
8	Loops	5	8	Debugging	5	8	Nested	5
9			9	While	5	9	loops	
10			10	If/Else	5	10	Unplugged	
11	Unplugged	0	11	Conditionals	5	11	Loops	5
12	Events	5	12			12	Conditionals	5
13			13			Unplugged	0	
			14	Unplugged	0	14	Event	5
			15	Build a game	5	15	Unplugged	0
						16	Function	5
						17		
						18		
	TOTAL	25		TOTAL	40		TOTAL	45

Table 56 Scores applied to Code.org courses by CS concept.

A scale of 5 points was chosen to enable further insight into children’s progress if required. The data gathering tool within the online resource provides a report on the progress children’ make within each lesson. The enables a scale of ‘5’ points for a child completing 100% of the CS concept, 4’ points for a child completing 75% of the CS concept lessons and activities, ‘3’ points for 50% of the CS concept lessons and activities and decreasing incrementally until the score of 1 which is awarded if the child did not complete the first CS concept that they started. Table 58 provides an overview of the scoring statements. However, this chapter reports on children’s completion of the CS concept and therefore the ‘5’ point marker was used. It should be noted that each lesson increases in complexity.

Score	Measure
5	Completed all lessons and activities for the CS concept
4	Completed 75% of lesson activities for the CS concept
3	Completed 50% of lesson activities for the CS concept
2	Completed 35% of lesson activities for the CS concept
1	Did not complete first lesson for the CS concept

Table 57 5 point scoring system

9.5.7.1 Children’s reading and numeracy attainment banding

Children’s reading and numeracy attainment in Class A and B is measured using CfE professional judgement levels. A score banding based on holistic assessment is awarded for individual children ranging through 1.1, 1.2, 1.3, 2.1, 2.2, 2.3. Each of the bandings in this school shows that the child has covered and achieved one sixth of the course for that year group. If a child in P5 has 1.1 then they have covered and achieved one sixth of the P5 course. At the time of the study in June, this would be well below the expected average. Most children should be 2.1 or 2.2. (Table 59) The same scoring system applies to Class B and the P6 coursework. If a child in P6 has a 1.1 banding they have covered 1/6 and achieved 1/6 of the course. Most children at P6 would be expected to be at 2.1 or 2.2.

Banding for CfE Reading and Numeracy teacher professional judgement

Banding	Covered and achieved	Average
1.1	1/6 of the course	Well below average
1.2	1/3 of the course	Average
1.3	½ of the course	Above average
2.1	2/3 of the P6 course	Below average
2.2	5/6 of P6 course	Average
2.3	Full course achieved	Above average

Table 58 Professional judgement of attainment in literacy and numeracy

Children’s enjoyment levels for toys games and activities survey

Data for analysis on relationships with children’s interests outwith formal education and their progress in introductory programming was gathered using an electronic survey(Add survey to the Appenidx). Using a Likert scale, scoring their enjoyment levels from ‘N/A’ (not appropriate or no experience) to ‘5 (Very high levels of enjoyment), children indicate their levels of enjoyment for 37 play activities.

For manageability, 37 activities, were collapsed to the same play categories used in quadrant C. The study acknowledges the challenges defining play and recognises its complexity. Every aspect of children’s development is a form of play all of which contribute to children’s physical, intellectual and socio-economic growth. Therefore, the activities are collapsed into; physical; pretence; object; symbolic and games with rules. In addition, an additional category of electronic toys was added as this type of activity was not present in Whitebreads and construction toys as these have been shown to relate to STEM spatial skills in previous studies.

For interrater reliability to collapse the play categories, the question categorisation was moderated by four early years experts for professional views. Table 60 shows the final groupings used in the data analysis.

PLAY THEME TOYS GAMES AND ACTIVITIES SURVEY QUESTIONS				
	QU*			
PHYSICAL	1.1	Climbing	1.9	Wrestling
	1.4	Riding bikes	1.33	Chasing friends
	1.5	Playing with balls	1.34	Physical play
	1.8	Trampolines	1.36	Dancing
OBJECT	1.6	Sand play	1.11	Play with stones, sticks or mud
	1.10	Water play	1.35	Play with objects
CONSTRUCTION	1.1	Building models	1.37	Lego
SYMBOLIC	1.12	Painting/drawing	1.18	Singing
	1.14	Reading on my own	1.19	Music
	1.15	Drawing	1.24	Play doh
	1.17	Being read to by others	1.28	Writing
PRETENCE	1.20	Dolls	1.23	Pretend shops
	1.21	Superheroes	1.25	Playing at schools
	1.22	Dressing up	1.32	Pretend play
RULE GAME	1.1	Card games	1.29	Team games
	1.15	Games with rules	1.30	Guessing games
	1.26	Board games	1.31	Playground games
TECH/COMP	1.7	Computer games		
	1.27	Electronic games		

Table 59 Questions from survey grouped by play categories

*Question number from the online survey children completed about their play interests.

This survey is amended from previous work looking at early development experiences and computing proficiency (Cutts, Patias et al 2017).

The table below shows each course selected by the teachers broken down by the lessons and activities to reinforce content. The columns show the CS content covered. Course C

has 13 lessons starting with unplugged activities through to Events. Each lesson has up to 14 activities. Course D has 15 lessons starting with unplugged activities and up to 15 activities for each lesson. Course E starts with unplugged activities, has 14 lessons with up to 14 activities for each lesson. Each course increases in complexity and there are examples where a CS content is repeated across a few lessons or revisited at a later stage. Loops are covered in Course C through lessons 5-6, then revisited in lessons 8-9 and again in Course E in lesson 5. Lessons with a predictive element are shown in column **P*/^**. **A lesson with an ‘*’** indicates that the predictive activity is at the end of the lesson. Lessons with ‘^’ indicate the predictive activity is at the beginning of the lesson.

NB: This table is extracted text from Code.org CS Fundamentals published lesson plans.

LESSON	COURSE C	P*/^	LESSON	COURSE D	P*/^	LESSON	COURSE E	P*/^
Class A			Class A - B			Class B		
1	Unplugged		1	Unplugged		1	Unplugged	
2	Sequencing		2	Sequence		2	Sequence	
3	Debugging - loops		3	Events		3	Unplugged	
4	Unplugged		4	Nested Loops	*	4	Debugging	
5	Loops	*	5			5	Loops	
6		*	6			6	Unplugged	
7	Unplugged		7	Unplugged		7	Loops	*
8	Loops	^*	8	Debugging	*	8	Nested loops	
9			9	While	*	9		
10			*	10	If/Else		10	Unplugged
11	Unplugged		11	Conditional		11	Loops	
12	Events		12			12	Conditionals	
13			13			13	Unplugged	
			14		Unplugged		14	Event
		15	Build a game		15	Unplugged		
					16	Function		
					17			
					18			

Column P*/^: * indicates that the prediction activity is at the end of the lesson. ‘^’ indicates the prediction activity is at the beginning.

Table 60 Breakdown of each lesson used in the study and the concepts covered.

Predicting what the program will do features in a few of the lessons at the end of the final activity. Children are shown a program and given multiple choice statements describing what might happen when the program runs. On a few occasions, the prediction type question is at the beginning of the lesson. An example of a predictive activity for loops within the Code.org lessons is shown in Figure 4. Taken from Course C, lesson 5 activity 12, children read the blocks and predict what will happen when the program runs from a multiple choice set of statements.

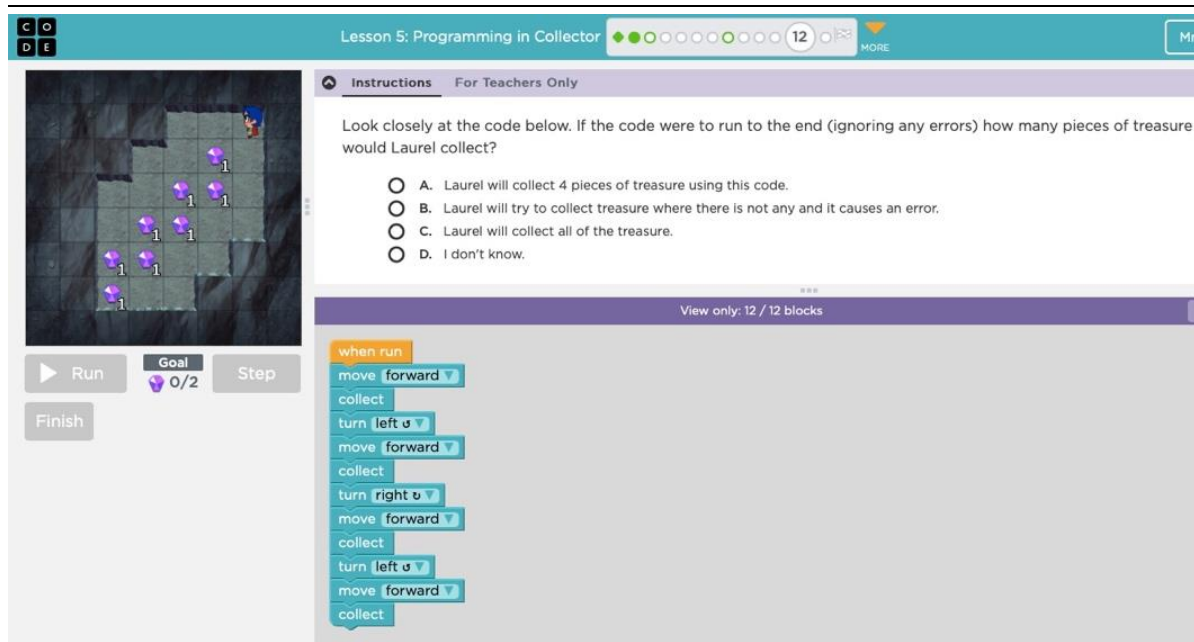


Figure 12 Screenshot of predictive element in course C Loops

Code.org Lesson and course incremental complexity.

Each course starts with basics and familiarises children with the environment. All programming takes place through dragging and dropping available blocks and connecting them into a coherent program. The output of the program is graphically illustrated in the form of animation through the themes of games such as ‘angry bird’ and ‘Frozen’. The aim of the resource in using ‘familiar’ themes is to add to its appeal to young children.

In the first few levels children move a character forward a few boxes or rotate. The lessons increase in difficulty with loops, conditionals and additional features. Each lesson increases in complexity; for example Course C lesson 4 loops is less challenging than Course C lessons 8-10 loops. In addition, Course D Events activities are much more complex than Course C lesson 12-12 Events. Course E loops are more challenging than Course C loops and course D includes functions with parameters and recursions.

Each Lesson has a series of consolidation exercises to reinforce teaching points within different contexts. Table 62 illustrates each of the context used for Courses C-E. Children may be taught ‘loops’ through Maze and later in another lesson with more challenging instructions through ‘Collector’. The aim of each theme is to consolidate learning and introduce complexity incrementally.








THEME	IMAGE	DESCRIPTION
MAZE		Catch the piggy and bring the zombie to the sunflower
COLLECTOR		Laurel the adventurer collects treasure
ARTIST		Create a script to draw a picture (similar to turtle and turtle graphics)
HARVESTER		The farmer needs to backfill holes and clear piles from the farm. (Extension of the maze context)
BOUNCE		Most games will involve sprites colliding with each other.
FARMER		The farmer has to get her field completely flat to start planting crops. Programs remove all the piles of dirt and fill in all the holes
FROZEN		Anna and Elsa explore the magic and beauty of ice. Programs create snowflakes and patterns as the sprite ice-skates making a winter wonderland that can be shared.

Table 61 Images from code.org showing content.

Each course reinforces learning through familiarisation within each 'theme'. Each lesson activity sets out a challenge for the user to complete using available blocks. The activity determines the number of blocks it should contain. Users can progress if they use more than the minimum blocks required to make the program run and achieve the desired outcome. There is an option to reveal a solution and enable children to move through the activity. This is not picked up by the online reporting system.

9.6 Results

The results of the study in phase one and phase two follow: the role of the teacher; Children's learning process; CS concepts covered; and, Predictors of success. Results describe observations of the teacher's roles during each phase and the learning processes of children. A detailed account of children's progress through the online learning resources Code.org⁷ is provided. Findings highlight the relationship between the predictor variables of gender, reading and numeracy with children's progress and 'comfort' levels through the introductory programming course. The report concludes with recommendations for further work in this area including continued collaboration with primary education and computing science education to improve outcomes for children in CS.

9.6.1 The Role of the teacher Phase one and Phase two.

Observation notes show that, in phase one, the teacher planned learning taking account of the CfE Computing Science national guidance. The course resource selected by the teacher is Code.org. All children in Class A start at the beginning of Course C and all children in Class B start at the beginning of Course D. There is no direct teaching input, children work through the activities individually. At the end of phase one children record their view of the level of difficulty using a 3-point scale. The scoring and measurements are outlined previously. Notes from observations show, teachers do not apply the Curriculum for Excellence assessment benchmarks and they omit unplugged activities.

In phase two, teachers use their typical approach to planned learning and implements a three-part lesson structure. They consider effective assessment for learning strategies such as 'Think Pair Share' and 'Show Me Boards'. They revisit the lesson activities and highlight tasks using prediction and show the relationship between how the program works and the blocks. They are given video clips created by the researcher of the programs running to facilitate discussions with children. The CfE 3 Step approach was discussed and

⁷ <https://studio.code.org/courses>

a flow chart outlining the hierarchical approach. The teacher's role in phase one and phase two taken from observation notes of the study is outlined in the table below.

Assessment for learning (AFL)

Assessment for learning known as AFL are strategies to improve children's performance and receive effective feedback (Black, and William, 2010). AFL strategies observed during the study are used by the teachers in the study in other areas of children's learning.

Children are familiar with these strategies. The detail of the AFL strategy is not included because that is outwith the scope of the study. However, it is worthwhile noting that the teachers in the study did not employ their typical learning and teaching strategies alongside the commercial resources lesson plans.

Table 63 shows the role of the teacher in each phase of the study. Each statement is recorded against the CfE 3-step approach and evidence of AFL strategies. In phase one, teachers planned and implemented learning taking account of CfE step 3 where children build the program. Teachers did not direct the children to the multi-level approach of CfE. There was no use made of AFL strategies in phase one.

In phase two, teachers planned and implemented learning taking account of all three levels of understanding from CfE. Classroom management included AFL strategies that children use in other areas of their learning.

ROLE OF TEACHER (summary from observation notes)	CfE step*	AFL
Phase One		
<p><i>Classroom management</i></p> <p>Plan learning using code.org and Identify children’s starting point.</p> <p>Facilitate children’s log on to the accessing course</p> <p>React to children’s queries</p>	3	No
Phase Two (INTERVENTION)		
<p><i>Classroom management (Primary Education effective practice)</i></p> <p>Organise children into 3 teaching groups – Peer instruction</p> <p>Identify teaching point for each group and 3-part lesson</p> <p>Formative assessment ‘Think-Pair-Share’ ‘Show me Boards’</p>	1,2,3	Yes
<p><i>Direct teaching group (15 min)</i></p> <p>Goal: CS concepts associated tools explored on different levels.</p> <p>Preparation: Lesson activities are reordered to facilitate looking at program as a whole; the relations between blocks; single expressions and predictions. (Typically, in code.org the predictions are at the end of lessons.)</p>	1-2	Yes
<p>Implementation: Show complete animations.</p> <p>Facilitate discussion of the processes observed in the virtual world animations and procedural thinking, relations between blocks and single expressions and how blocks, relate to the animation processes.</p>		Yes
<p>(Single expressions or instructions to blocks, relations to blocks and the whole program (reverse order) Through Peer Instruction ‘Think pair share and show me boards’-Illicit children’s understanding of the blocks and associated actions of the animations.</p>	1-2	Yes
<p>Quiz:</p> <p>Run scripts for each concept. Using multiple choice children predict the outcomes explaining their method of problem solving.</p> <p>Run scripts to validate</p>		Yes

Organise children in teaching groups and direct them to the onscreen lessons reinforcing the teaching point. Children engage in peer support. Support children during on screen activity queries using procedural thinking.	3	Yes
Plenary Reinforce the relationship between the animations and the blocks.		Yes

Table 62 Role of the teacher from observation notes

**Curriculum for Excellence CS Experiences and Outcomes organisers (Curriculum for Excellence Technologies 2017)*

The children’s learning process

Analysis of the observation notes show that during phase one children are expected to; read; understand and follow all online instructions. Teachers support children on an ‘ad-hoc’ basis. Phase two makes explicit the CfE 3-step comprehension-first approach and applied it to the resources.

Observations from notes of children’s actions and cognition (mental processes) in phase one and phase two show are as follows. In phase one, children create new programs by dragging available blocks to either complete or modify a partial program. In phase two, the intervention involves children working through a sequence of steps using procedural language during ‘Think Pair Share’ and plenary sessions. The whole process children are involved in links to Shulte’s (2008) education model of program comprehension (block model). The focus is reading and program understanding instead of the focus in phase one which is to create a script straight away. describe the program (script) as a whole, then its execution and finally read the blocks word by word at text surface level to construct meaning. This is similar to how a child processes language when learning to read. They hear the spoken word assigned to an object or activity, make connections with the written word associated with the object or activity before writing.

Step one, children interpret an animation using procedural language describing the processes that they observe. Individually and in pairs they describe the series of well-structured steps and procedures within the script reading each of the blocks. They develop and demonstrate a level of understanding about each block, the functions and commands.

Step two, children zoom in from discussing whole program output as a whole and connect the single expressions or instruction blocks to the program function from their observations. To reinforce further children's understanding of individual blocks, they continue to read and discuss each block. This is known as understanding at text surface within Shulte's (2010) Block model. They continue to discuss and develop an understanding of how the blocks individually and collectively contribute to executing the script or simple program. Children spend time predicting the behaviour of the Script. Moving to step 3, children continue to reinforce their understanding of the blocks and the program through modifying existing programs within the activities. They complete partial programs predetermined by the activity.

The table below provides an overview of children's cognition and mental processes observed during phase one and phase two. The 'Phase One' column represents the create-first approach where children immediately build scripts or simple programs. All actions that children undertake relate to CfE step 3 where children create solutions using the available blocks in the online activities. The activities vary in expectations, although children repeat the same process which is either creating a program from pre-defined blocks, completing a partially written program or modifying segments of code. In a small number of examples, children predict the behaviour of a given program. They do not cover CfE step one or step two.

The Phase two column shows children using the comprehension-first approach. They explore the program on several levels before building. This multilevel analysis of the program takes children's learning through the CfE three steps. They begin by observing and describing the processes taking place in the animation. By understanding the processes, they make connections to the program blocks and in each lesson they are expected to predict the behaviour of the program before starting the activities. Prediction is included in all of code.org CS Fundamentals course lessons and the materials for the intervention were adapted to apply this approach in each lesson. Completed scripts were recorded and shared with the children inviting them to predict what would happen next. The next two learning processes repeat phase one and children modify a program or complete a program as directed by the activity. In phase two they also discuss the programs they create and describe what is happening either with their peers or the teacher who moderates their understanding.

Phase 1	Phase 2
Complete a partial program in order to achieve an output and reinforce a concept.	Describe the process observed in the real/virtual world.
Modify segments of code to complete a program with a given goal.	Connect blocks to the processes observed.
Create a program from pre-defined blocks when the goal of the program is given.	Predict the behaviour of a given program.
Predict the behaviour of a given program	Modify a program in order to achieve a given output.
	Complete a partial program in order to achieve a given output
	Create and describe programs accurately using computational tools.

Table 63 Children's mental processes during phase one and phase two

9.6.2 Children’s view of the level of difficulty (Comfort Level)

Children rated their comfort level at the end of phase one and phase two of the introductory programming course. The traffic light system participants use in other areas of their learning was applied at the end of phase one and phase two.

Score (1) if children found the learning “Hard”, they would use a red circle and this informs the teacher that; “I found this hard or tricky, I don’t understand it yet”.

Score (2) If children found the learning “Ok” : they would use a yellow circle and this informs the teacher that; ““I think that I could understand it, but I could not explain it to someone else”

Score (3) If children found the learning “Easy” : they would use a green circle and this informs the teacher that; “I found this easy and I could explain it to a friend”

All children report phase one as more difficult than phase two. In Class A the level of difficulty phase 1 mean rises from 1.8 to 2.8 in phase 2. In class B the level of difficulty mean in phase 1 rises from 1.4 to 2.8 in phase 2. The exception being two children in Class A phase one who reported phase one as ‘easy’ they also reported phase two as ‘easy’.

Progress and comfort levels : All children report less difficulty in phase two and their progress scores improve.

	Course C-D Class A			Course D-E Class B			
	MIN	MAX	MEAN		MIN	MAX	MEAN
Phase 1	1.00	3.00	1.8		1.00	2.00	1.4
Phase 2	2.00	3.00	2.8		2.00	3.00	2.8
Children’s mean of their comfort level in Phase 1 and Phase 2							
*Level of difficulty reported by children 1=“hard” 2=“ok” 3=“Easy”							

Table 64 Children’s comfort levels in introductory programming

9.6.3 Computing science concepts covered

The teachers used code.org as the main resource for the introductory programming course.

Code.org resource

In Phase one, the teachers selected Code.org as the main resource to teach the concepts. The resource is the main context for learning and teacher's planning notes align the commercially produced resource to the Scottish Curriculum for Excellence computing science experience and outcomes. Code.org lessons increase in complexity through each of the activities, lessons and courses.

Code.org is a visual programming environment aimed at children age 4 upwards. The content is available online with no financial cost. It is a web application and not platform dependent and children access the resource using an account which tracks their progress. The 'programming' element to learning CS concepts, involves children dragging and dropping available blocks into a coherent programme creating an animation similar to a 'gaming' environment. Each course is aimed at an age group determined by the USA schooling system. The teachers in the study chose CS Fundamentals (2017) Course C – 2nd grade (age 7-8), Course D – 3rd grade (age 8-9) and Course E – 4th grade (age 9-10).

Figure 12 is extracted from the code.org teacher planning materials. It shows that the CS Fundamentals course selected by the teachers is aimed at the USA schooling system from k-5.

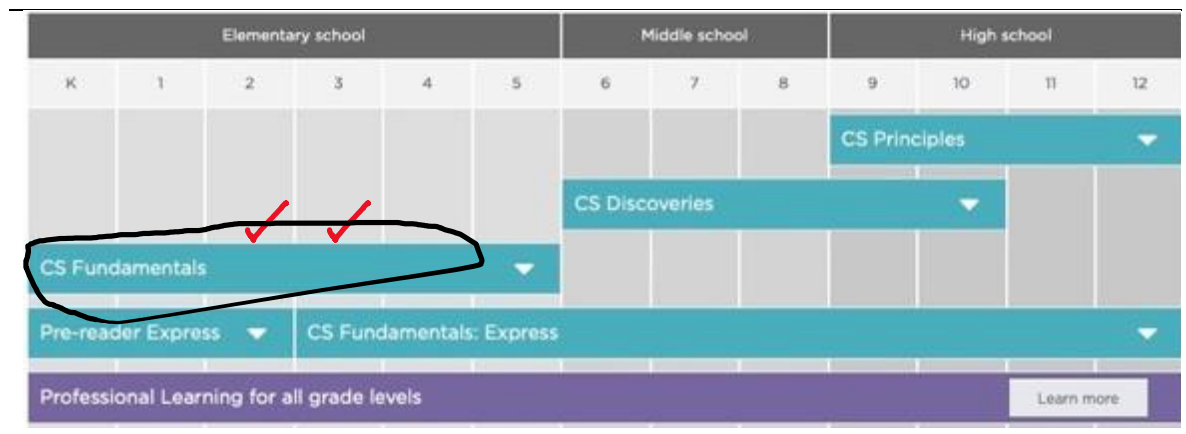


Figure 13 Code.org lessons matched to USA school system

Children's progress covering each computing science concept is measured using a 5 point scale.

CS concept – Children's progress:

- 1) Class A scores in phase one range from 10-45 with a mean of 27.5 rising to a score range 25-85 in phase two with a mean of 56.5.
- 2) Class B scores in phase one range of 5-30 with a mean of 17.3 in phase 1 changes to 15-60 with a mean of 36.2. (Table 66)

	Course C-D Class A			Course D-E Class B		
	MIN	MAX	MEAN	MIN	MAX	MEAN
Scores						
Phase 1	10.00	45.00	27.5	5.00	30.00	17.3
Phase 2	25.00	85.00	56.5	15.00	60.00	36.2

Table 1: Children’s progress mean and their view of difficulty in Phase 1 and Phase 2

*Level of difficulty reported by children 1=“hard” 2=“ok” 3=“Easy”

Table 65 Children’s progress through phase one and phase two

The table below shows the progress children in Class A make through the CS concepts in phase one and phase two. Each CS concept must be completed before moving on to the next one. Class A progress is evident after the three week intervention. It should be noted that children spend half the time working as individuals on the computer in phase two. Half the lesson is group discussion with much less screen time.

In phase two, class A scores increase with 4 more children completing loops than in phase one. An additional 11 children complete Loops (2) taking the total to 24. There are rises in phase 2 of 18 more children completing sequencing revision and 20 children completing events. 1 child made very good progress after the intervention was applied moving through the completing If/else. The table is limited because it is not clear the progress for individuals from phase one to phase two and that would be worthwhile for future work.

CLASS A COURSE C						CLASS A COURSE D						
Phase	sequence	Debug loops	loops	Loops (2)	Events	seq revision	events	nested loops	debugging	While	If\Else	Conditional
P1	26	26	22	13	11	2						
P2	26	26	26	24	24	20	20	5			1	

Table 66 Total no of chn. in Class A completing CS concepts in phase 1 and phase 2.

Class B progress

Class B progress is similar to class A with more children completing the increasingly challenging lessons. In phase one the teacher planned all children to start at course D irrespective of their prior knowledge. 2 children made very good progress in phase 2 with the intervention covering the remainder of course D and 6 lessons within course E.

	CLASS B			COURSE D					COURSE E								
Phase(P)	seq revision	events	nested loops	debugging	While	If\Else	Conditional	Build a	Seq	Debugging	loops	Loops (2)	Nested	Loops (3)	Conditionals	Event	Function
P1	28	28	21	21	21	14	8	4	2								
P2	28	28	28	28	28	27	27	22	16	13	13	13	6	3	3	2	2

Table 67 Total number of children in Class completing CS concepts.

9.6.4 Predictors of success

Predicting outcomes in introductory programming is a focus of the CSED research community. Single factors of age, gender, background and math are core variables for deliberation. This section explores the following variables in relation to children's IP courses in an authentic primary classroom:

- **Reading**
- **Numeracy**
- **Interests in toys games and activities**

This exploration is set within the framework of identifying possible indicators of success in primary school IP courses in formal education with a view to inform future work. The following questions relate to the overarching research questions.

Does children's reading ability affect progress in phase one or phase two?

Does children's numeracy ability affect progress in phase one or phase two?

What level of difficulty do children report when learning CS concepts in phase one and phase two?

Does children's reading ability affect their reported levels of difficulty in learning CS concepts?

Does age, gender, reading attainment, numeracy attainment correlate with children's progress during introductory programming?

Do children's enjoyment levels of play correlate with their progress in introductory programming

9.6.4.1 Does children's reading ability affect progress in phase one or phase two?

Class A: 5/8 children with 'well below average' (Banding 1.1) reading attainment made the least progress in phase one. Observation notes show that the 1 child with n= 2.2 'above average' reading attainment who made the least progress in phase one was absent on one lesson and the computer that was used had technical difficulties in phase one. The 2 children making the most progress in phase one have 'above average' reading attainment. In phase two all scores improve. 4 children 'well below average' make accelerated progress and achieve as well as children with higher reading ability. In phase 2, progress clusters emerge, and 15 children achieve the same scores.

Class B: In phase 1, '5' children 'well below average' n= 2 and 'below average' n= 3 make the least progress in phase one. In phase two all children make better progress. The children with 'above average' reading attainment n=2 make the most progress in phase one and phase two.

In phase 1 of the 8 children with 1.1 (well below average) reading banding a total of 5 children completed the lessons and activities on 'sequencing and debugging loops; 2 children completed sequencing, debugging loops and loops; 1 child completed sequencing, debugging loops, loops, loops (2) and events. Data from this same cohort of children shows at the end of phase 2, 1 child completed sequencing, debugging loops and loops; 2 children completed sequencing, debugging loops, loops, loops 2 and events. 4 children completed sequencing, debugging loops, loops, loops 2 and events in Course C and revision of sequences, events, nested loops in Course D; 1 child completed all of Course C and revision of sequences, events, nested loops, debugging, while, If/Else and conditionals in Course D.

Table 69 shows the number of children in each reading banding and their progress in phase 1 (P1) and phase 2 (P2). The lowest reading banding is 1.1. Each child is represented by their reading attainment banding. For example, in phase 1, 5 children with below average reading bands of 1.1 completed debug loops.

CLASS A – COURSE C – D

COURSE C

COURSE D

Read attain	Phase(P)	seq n	Debug loops	loops	Loops (2)	Events	seq revision	events	nested loops	debugging	While	If\Else	Conditional	Build a
1.1	P1		5	2		1								
	P2			1		2			4				1	
1.2	P1				1									
	P2								1					
1.3	P1			1		4								
	P2							1	4					
2.1	P1			4	1	2								
	P2						4		3					
2.2	P1	1		2		2	2							
	P2			1			1		3	2				
Tot a	P1													
Tot a	P2													

Table 68 Class A Reading attainment groupings and progress in phase 1&2

*1.1 (Well below average) 1.2 (Below average) 1.3 (Below average) 2.1 (Average) 2.2 (Above average)

CLASS B COURSE D

COURSE E

Rea	Phase(P)	seq	vision	events	nested loops	debugging	White	IfElse	Conditional	Basic game	Assertion	Debugging	loops	Loops(2)	Nested loops	Loops(3)	Conditionals	Event	Function
1.1	P1			2			2	1											
	P2						2			1	1							1	
1.2	P1			2															
	P2						2												
1.3	P1			1															
	P2						1												
2.1	P1						4	1	1										
	P2									2	1				1			2	
2.2	P1						1	4	3	2	2								
	P2								1	3				3			3	2	
Total	P1			5			7	6	4	2	2								
Total	P2						5			4	5				4			6	2

CS concepts covered by children grouped by reading attainment

*1.1 (Well below average) 1.2 (Below average) 1.3 (Below average) 2.1 (Average) 2.2 (Above average)

Table 69 Class B Reading attainment and progress

9.6.4.2 Does children’s numeracy ability affect progress in phase one or phase two?

Class A: Numeracy attainment data was the same as literacy and therefore findings are the same.

Class A: In *phase one*, $\frac{3}{4}$ children with the lowest numeracy attainment ‘well below average’ make the least progress. Children with a range of numeracy abilities from ‘below average’ to ‘above average’ make the next level of progress. 2 children with the highest numeracy attainment make the most progress in phase one and phase two. In *phase two*, progress accelerates for all children. The 5 lowest achieving children in phase one are the lowest achieving in *phase two*. However, the course content covered in phase two is more complex. (Table 71)

Numbers of children grouped by reading attainment and CS concepts covered.

CLASS B – COURSE D – E

Read attain	Phase(P)	COURSE D								COURSE E								
		seq revision	events	nested loops	debugging	While	If\Else	Conditional	Build a game	Seq (revision)	Debugging	loops	Loops (2)	Nested loops	Loops (3)	Conditionals	Event	Function
1.1	P1		3			1											1	
	P2					3												
1.2	P1		1															
	P2					1												
1.3	P1		1			3												
	P2					1			2			1						
2.1	P1					1	1	1	1									
	P2											1	1				2	
2.2	P1					3	4	3	1	2								
	P2								2			3	3				3	2
	Tot P1		5			7	5	4	2	2								
	Tot P2					5			4			5	4				6	2
		CS concepts covered by children grouped by reading attainment																
		*1.1 (Well below average) 1.2 (Below average) 1.3 (Below average) 2.1 (Average) 2.2 (Above average)																

Table 70 Class B Numeracy attainment of children and progress

9.6.4.3 Children's reading attainment banding and their comfort levels in learning CS concepts.

As mentioned in the learning process, each child records their comfort levels with the introductory programming course at the end of phase one and at the end of phase two using the traffic light system. A score is applied to each statement.

Children's view of level of difficulty and their reading attainment

All children report that their comfort level improves in phase two. In both class A and class B, 10 children who find the course hard have a range of reading bandings:

7 children who find the course hard are 'well below' average for their reading attainment. 3 children who find the course hard are 'above average' from their reading attainment. In class 4 children who are 'above average' in their reading attainment find the phase one course 'Easy'.

1 child out of 8 children with the lowest attainment reading banding found the course 'OK'.

In phase two, children's comfort levels improve with 2 out of 8 children with the lowest reading attainment banding 1.1 (well below average) finding the course 'OK' and think that they understand it. The remaining 6 children with the lowest reading attainment find phase two 'Easy' and that they could explain it to a friend. (Table 72). The level of difficulty and reading attainment is worth noting because the approach is program comprehension.

These results show clearly that all children find by the end of phase two that their view of the level of difficulty has improved. Improved levels of difficulty could be attributed to the longer time children spend on the course. However, the marked improvement in children with the lowest reading attainment and their view of the level of difficulty is worth considering in future work. The intervention is a comprehension-oriented approach and reading is linked to comprehension. It is possible that using similar strategies to support children read when applied to visual programming IP courses does work.

Importantly, by the end of the course children do view IP positively and that they could share their understanding with a friend.

Read lev*	Class A reported level of difficulty						Class B reported level of difficulty					
	Phase 1			Phase 2			Phase 1			Phase 2		
	Hard	Ok	Easy	Hard	Ok	Easy	Hard	Ok	Easy	Hard	Ok	Easy
1.1	7	1			2	6	3	1	1	0	1	4
1.2	1			0	0	1	0	1	1	0	0	2
1.3	1	4	0	0	0	5	1	0	0	0	1	0
2.1	5	2	0	0	1	6	2	4	0	0	2	4
2.2	3	4	0	0	2	5	4	4	4	0	1	11

*Reading attainment *1.1 (Well below ave) 1.2 (Below ave) 1.3 (Below ave) 2.1 (Ave) 2.2 (Above ave)*

Table 71 Children’s comfort levels and reading attainment

Does age, gender, reading attainment, numeracy attainment correlate with children’s progress during introductory programming?

To explore children’s age, gender, reading attainment and numeracy attainment with their progress, Table 73 shows a correlation matrix showing the Pearson coefficients between different variables in a data set. This is a measure between two variables and was used to see if there was a linear relationship showing that one score affects the other and to estimate the strength. Reading Attainment and Numeracy Attainment are continuous variables. This is the most appropriate approach for this type of work. There are limitations using this approach for gender because gender cannot be ranked. Therefore gender has been included but not considered. These results show a degree of association and not causation.

- In class A, children’s reading and numeracy attainment and children’s course progress is statistically significant. ($n = .68^{**}$) in phase one, and not in phase two.

This suggests that in this particular study the higher the reading attainment in phase one the better progress that children make. Children’s reading and numeracy attainment was less of a predictor of progress after the intervention was applied. This change could be attributed to children benefitting from the teacher in phase two spending time explaining the approach whereas in phase one children worked independently reading the on screen instructions. Although the study cannot claim that the intervention mitigates reading and

numeracy attainment as a predictor, it does reflect the same pattern of results that was found in school B and is worth exploring in future work.

- In class B, children’s reading attainment (RA) and numeracy attainment (NA) is statistically significant in phase one and phase two. RA, phase one progress $n=.73^{**}$ phase two progress $n=.60^{**}$ NA, phase one progress $n=.74^{**}$, phase two progress $n=.64^{**}$

Class B results show that children’s reading and numeracy attainment levels and their progress in phase one and phase two remain the same. It suggests that the intervention did not impact on the predictors of success. From observations there is no clear explanation of why this occurred. Further work is required to explore the impact of the intervention on the predictor variables of reading attainment and numeracy attainment.

Table 8 Age, gender, reading attainment, numeracy attainment and children’s progress correlation

	Class A (Course C_D)						Class B (Course D_E)					
	Age	Gen	RA	NA	Prog-P1	Prog-P2	Age	Gen	RA	NA	ProgP1	ProgP2
Age	1	0.25	0.06	0.14	0.14	0.29	1	0.13	-.60**	-.55**	-.49*	-.44*
Gen	0.25	1	0.03	0.17	.46*	.51**	0.13	1	-0.02	-0.08	0.15	0.26
(RA)†	0.06	0.03	1	.86**	.68**	0.33	-.60**	-0.02	1	.93**	.73**	.60**
(NA)‡	0.14	0.17	.86**	1	.68**	0.34	-.55**	-0.08	.93**	1	.74**	.64**
Prog(P1)	0.14	.46*	.68**	.68**	1	.62**	-.49*	0.15	.73**	.74**	1	.90**
Prog(P2)	0.29	.51**	0.33	0.34	.62**	1	-.44*	0.26	.60**	.64**	.90**	1
Age (Years)	Gen(der) M=1 F=0						†RA Reading attainment					
**Correlation is significant at the 0.01 level (2-tailed).												
*Correlation is significant at the 0.05 level (2-tailed).												
‡NA Numeracy Attainment												

Table 72 Reading and numeracy attainment and course progress

9.6.5 Children’s reported levels of enjoyment in toys, games and activities.

Block play provides a context for exploring the world and developing skills in STEM. Research shows that young children often engage in engineering-based design to solve problems through spatial language and social behaviour using construction toys. Using a survey this study looked at the full range of participants play interests and not just block play for direct relationships with children’s success in introductory programming. Data gathered from children’s survey results was collapsed using Whitebread’s play categories linked to cognitive development. An additional category of digital/technology category is included.

Given the evidence for construction toys and STEM skills a separate analysis was undertaken to show correlations.

9.6.5.1 Do play interests correlate with children's progress in introductory programming?

Progress in phase one and phase two

Children's play interest and progress in the phase one and phase two was measured using the Pearson correlation coefficient between different variables in a dataset. The Pearson correlation coefficient is a measure of the linear association between two variables. In summary, the correlation is significant at the 0.01 level (2-tailed) in both class A and Class B progress in phase one and children's progress in phase two (Table 72):

Children's progress and their play interests

The correlation matrix outputs show that there is statistical significance between children's reported enjoyment levels in a few play activities and their progress in phase one and phase two:

- Class A shows that there is statistical significance with children's interests in physical, object and construction; symbolic, object and pretend; pretend play and rules and progress across the two phases.
- Class B shows that there is statistical significance with children's interests in object and construction, pretend and symbolic.

The exploratory results show an association between some play interests and enjoyment levels. However, study limitations mean these findings cannot be generalised to a wider population.

Symbolic and pretend play interests are highlighted across both classes. It is known that playing with blocks is a central activity for young children. While there are challenges making strong claims based on the data, it is worth acknowledging that these results align with the emerging literature insights on block play and STEM success. (Tunnicliff, 2022). However, it is also worth noting that how one child plays with blocks can be different from another across the three domains of:

- Sensorimotor
- Symbolic
- Construction (Wolfgang, and Stannard, et al, 2001)

Therefore, future work could explore further the relationship of play interests with a view to considering early play activities and a developmentally appropriate CS curricula for the youngest of learners before they start school.

CLASS A

		Course C-D							
		Phase		MEAN OF PLAY GROUPINGS					
	1	2	Physical	Object	Symbolic	Digital	Rules	Pretend	Construct
PHASE 1	1	.62**	0.19	0.18	0.05	0.26	0.05	-0.12	0.34
PHASE 2	.62**	1	0.07	-0.06	-0.15	0.12	-0.21	-0.21	0.03
PHYSICAL	0.19	0.07	1	.72**	.74**	0.15	.47*	.56**	0.33
OBJECT	0.18	-0.06	.72**	1	.48**	0.20	.46*	.39*	.72**
SYMBOLIC	0.05	-0.15	.74**	.48**	1	- 0.09	.41*	.60**	0.20
DIGITAL	0.26	0.12	0.15	0.20	-0.09	1	0.16	0.05	.49**
RULES	0.05	-0.21	.47*	.46*	.41*	0.16	1	.57**	0.26
PRETEND	-0.12	-0.22	.56**	.39*	.60**	0.05	.57**	1	0.15
CONSTRUCTION	0.34	0.023	0.33	.72**	0.20	.49**	0.26	0.15	1

**Correlation is significant at the 0.01 level (2-tailed).

*Correlation is significant at the 0.05 level (2-tailed).

Table 73 Correlation matrix progress and levels of enjoyment in play

CLASS B

Course D-E									
Progress		MEAN OF PLAY GROUPINGS							
	1	2	Physical	Object	Symbolic	Digital	Rules	Pretend	Construction
PHASE 1	1.00	.90**	0.10	0.00	-0.11	0.05	0.18	-0.32	-0.03
PHASE 2	.90**	1.00	0.05	-0.03	-0.22	0.09	0.17	-0.35	0.03
PHYSICAL	0.10	0.05	1.00	.48*	0.37	0.30	0.08	0.17	0.15
OBJECT	0.00	-0.03	.48*	1.00	.420*	0.03	.49*	.46*	.74**
SYMBOLIC	-0.11	-0.22	0.37	.420*	1.00	-0.16	0.21	.55**	-0.06
DIGITAL	0.05	0.09	0.30	0.03	-0.16	1.00	0.35	-.40*	0.30
RULES	0.18	0.17	0.08	.49*	0.21	0.35	1.00	0.25	.47*
PRETEND	-0.32	-0.35	0.17	.46*	.55**	-.40*	0.25	1.00	0.14
CONSTRUCTIO N	-0.03	0.03	0.15	.74**	-0.06	0.30	.47*	0.14	1.00

***Correlation is significant at the 0.01 level (2-tailed).*

**Correlation is significant at the 0.05 level (2-tailed).*

Table 74 Correlation children’s progress and level of enjoyment in play categories

9.7 Discussion

In an attempt to negate the well-documented CSED HEI issues of high failure rates, (Margulieux, and Morrisson, 2019) cognitive load, misconceptions and materials that are too challenging, introductory programming in primary school needs well-considered implementation (Salac, 2019). Visual programming tools readily available online at little or no cost may offer an instant solution for the primary classroom. This exploratory study highlights potential issues with implementation of a computing science course planned in accordance with the Scottish national computing science curriculum. The study consists of both quantitative and qualitative data. The qualitative data brings further insight to the quantitative data and the progress that the children make in introductory programming.

In phase one teachers planned independently of the researcher and children with lower reading attainment made less progress than their peers. In addition, almost all children reported phase one to be ‘hard’ or ‘ok’. In phase two, using an approach that applied the Scottish 3-Step approach to the phase one materials, children’s progress accelerated, and more children viewed the materials as easy and could share their knowledge with peers. Interestingly, the teachers needed prompting to apply typical effective learning and teaching strategies that they use in delivering other curricular areas. Perhaps because the commercially produced materials in the study are focused primarily on the lesson content and not the delivery. However, after being prompted they were able to plan the comprehension-first using typical learning and teaching strategies.

9.7.1 The ‘Role of the teacher’

The teacher’s role in phase one provides minimal direct teaching instruction and children working independently through the code.org materials. While this is not recommended by the resource itself, the practice is possibly not dissimilar when teachers are faced with a new subject to deliver. The teachers in phase one both undertook a management role as opposed to leading learning. In the second phase not only did the teachers follow the ‘3-step’ process more closely, when prompted, they naturally included typical pedagogical approaches used in other curricular areas. Children were grouped according to their progress, the 15-minute teaching input prior to working independently meant that the teacher had to explore the materials prior to the session, identify the prediction element in the course and use this to improve their own understanding and the children’s. In addition, by introducing a plenary session again that was typical from their work in other curricular

areas teachers found that they had an opportunity to look ahead to what was coming next. Minimal scaffolding by the author was required to increase the primary teachers' understanding of the CS concepts and the associated blocks. In addition, they were able to apply the comprehension-first approach and recognised its relevance. Independently, one teacher made a link with her understanding of early literacy and the impact of listening and talking before reading before children are expected to write.

“Ah, this makes perfect sense, it's the same as how we teach the children to read” Teacher A School B (Quadrant D)

Waite, J., (2023) in her paper *Designing in K-5 projects recognising that teaching programming is more than just teaching the syntax of programming language*. She proposed that novices should be taught how to create programs, including how to design them. This approach also replicates teaching of writing in primary schools where children use writing frames to construct a story. This scaffolding approach appears to support children and teachers well and could be part of the primary school teacher's 'toolbox' after children gain experience listening, talking, reading and understanding as shown in this thesis.

9.7.2 The children's learning process

In phase one, children work in isolation as they moved through lessons self-learning. The majority found this approach 'hard' or 'ok' by comparison to their view of the level of difficulty in phase two when the tools embodying the CS concepts are emphasised through facilitated discussions by the teacher, peer support and there is also a focus on understanding the processes taking place in the program implementation and the associated CS vocabulary. Although some children did not accelerate their progress in the second phase to achieve the same as or overtake their peers, they did progress and importantly their view of the level of difficulty notably improved and their perceived level of difficulty reduced.

Understanding the multilevels of programming is an approach used in tertiary education (Schulte, 2008) and this study shows its appropriateness for younger children. The study takes account of children's age and stage of development while applying key learning through analysing the tools that embody CS concepts. As a result, children experience a breadth of CSED as expected by the Scottish curriculum within the same time available as the more narrow approach in phase one of the study where teachers plan and implement a course independently.

9.7.3 The CS Concepts

The study shows that children's progress improves in phase 2 and more children indicate that they understand the CS concepts and can explain their learning to a friend.

Importantly, this intervention enables coverage of the breadth of the Scottish curriculum experiences and outcomes. The intervention enables children to build the foundations of programming which will support their progression through to secondary education where at a later stage they may choose to specialise in the subject.

9.7.4 Predictive variables

The improvement in children's progress through the CS concepts with low reading attainment bandings is an important consideration. The findings support the view that facilitated discussions looking at the different levels of the scripts deepens children's understanding of the blocks graphical names. They can read, interpret the blocks knowing the actions and functions how they contribute to the script as a whole. This approach in this study disrupted the pattern of low achievement recorded in phase one for this group of children.

Findings show gender issues and children's progress, in class B males achieve more than females in both phase one and phase two. However, there are no gender issue with children's reported comfort level.

9.7.5 Added value of the intervention

In Phase one, observations show children's learning processes relate to CfE step 3 and they create computational solutions. The learning process listed in Phase two, with the intervention, shows that children use multilevel exploration of the program. They discuss their understanding of the program with peers or their teacher.

Findings show that children make better progress when they are shown the program as a whole and asked to predict outcomes from code.org programs before step 3 and creating scripts. This improvement is because children make connections with the blocks and their functions. They improve their understanding of the block names as well as their functions and how they contribute to the program goal as a whole. The children with 'well below' reading ability in phase one are expected to read and follow instructions in a new area of learning and not surprisingly they perform less well.

Code.org has a plethora of resources to support teachers implementing the courses. Phase two identified where the predictive element was in each lesson and used that as the teaching tool displayed on the large screen for group discussion. This means minimal resource implications for teachers looking to implement the course using CSED methods.

This research describes the teacher's role, the children's learning process, the CS concepts and predictive variables using Code.org in an introductory programming course in a primary school. It answers the research questions:

RQ 2: What typical programming approaches are in place in tradition primary classrooms and how successful are they?

Primary teachers in the study used commercially produced resources when planning and implementing introductory programming to primary school age children. The resources employed a create-first approach that teachers embed within their typical classroom management and routines.

RQ 3: Can a comprehension-first oriented programming course be implemented successfully in a primary school classroom?

Working in collaboration with primary school teachers, a comprehension-first approach can be applied to their existing classroom instructional design processes to maintain familiar routines for children. There are benefits in planning listening and talking strategies within the learning for children to understand the processes that they will represent through code. This process is similar to the teaching of children's writing in early literacy where they spend considerable time listening and talking before reading and then writing.

The intervention takes account of primary teachers' zone of proximal development and scaffolds their knowledge. It brings together their primary learning and teaching practice with CSED research. The approach facilitates children's discussions enabling them to reinforce their understanding of the tools that use the CS concepts before creating digital solutions. Prior to applying the intervention children with low reading ability banding are negatively impacted. In phase two their progress levels and comfort levels improve. However, there is an acknowledgement that further work is required with particular interest on gender.

9.7.6 **Study limitations**

The study is an exploration and applies the intervention to the same cohort of children. Further work is required through quasi-experimental design to determine the effects of the intervention in a controlled environment. Data analysis in school B does suggest that improvements are not attributed solely to more time being spent on the activities. However, it does show that using the 3-step approach children's comfort levels improve, and the breadth of CS experiences and outcomes are in line with national expectation. An assumption is that without the intervention, children continue to create solutions without opportunities to establish their understanding of the CS concepts.

Chapter 10

Discussion/Future work

Thesis statement:

‘Recognising the importance and challenge of introductory programming for all primary pupils, there is value in balancing CS education research alongside primary teacher expertise in a comprehension-first pedagogy.’

Research questions

RQ1: Is there value in computing science education for all?

RQ2: What typical programming approaches are in place in traditional primary classrooms and how successful are they?

RQ3: Can a comprehension-first oriented introductory programming course be implemented successfully in a primary school classroom??

10.1 Summary

Over recent years computing science is featuring globally in formal primary schools education and this thesis aims to explore its value in primary school education and how primary school teachers plan and implement the new subject. Another goal was to explore a comprehension based approach in a traditional classroom setting.

Due to the evolutionary nature of policy implementation, the thesis employed an iterative approach. This iterative approach allows for content and methodology to be adapted over the course of the research (2015 – 2018). It enables learning from one study to inform the next (Bickman and Rog 2008):

‘An iterative approach is that research questions can be changed over time based on material collected and that strategies, data collection and analysis methods and tactics should fit the changing research questions and process phases’

The study is organised into interconnected quadrants each containing a related work literature section and a study.

Quadrant B (2016) contributes to RQ2. It explores approaches to primary school introductory programming in traditional classroom settings. Background reading shows that most literature at this time is theory or content driven with few data driven studies. In addition, studies are led by the CS research community with no insights into primary teachers own adoption of CS in a typical classroom setting. The study in four schools found that educators use visual programming languages and employ a create-first approach. In addition, a few children are demotivated by the experience. This study highlights some of the issues with CS in tertiary education contexts are beginning to appear in primary school. Future work suggests a study using CS research informed practice.

Quadrant C (2017) contributes to in part RQ 2 and RQ3. Scotland refreshes the CS curriculum underpinned by a comprehension-first oriented approach. At this stage, few CS experts would have the knowledge of primary school classrooms and few primary school teachers would have the CS expertise. Therefore, a modified comprehension-first oriented approach was planned and implemented by the Ambassadors with no primary school experience in 2 schools with 6 cross age classes involved. Materials were moderated by a CSED expert and the lessons were observed by the author of the thesis who has considerable experience in primary education and the classroom teachers. This enabled an evaluation of the approach in a primary school classroom. Findings show that the comprehension-first oriented approach motivated children and they made good progress. However, any approach needs to be replicable in a primary school classroom and future work suggests a collaboration with the CSED comprehension-first approach and primary education experts.

Quadrant D (2018) contributes to in part RQ2 and RQ3. A comprehension-first approach for formal school education is beginning to emerge in literature. However, the approach continues to be based on tertiary or secondary education contexts. This study therefore is set in two contrasting primary school contexts. It involves four experienced primary school teachers planning and implementing a 3 x 1 hour introductory programming (phase 1) course. The author observes the lessons and in collaboration with the primary teachers a further 3x1 hour course (phase 2) is planned and implemented using a comprehension-first approach. Findings show that in phase one, teachers continue to use visual programming environments using a create-first approach. In phase two, children cover the CS curriculum in greater depth and show higher levels of motivation. Importantly, the teachers make a connection with the comprehension first oriented approach and how they teach

literacy. The intervention was applied taking full account of a primary teacher's classroom management and routines.

The thesis concludes with a brief overview of course content coverage in quadrant B, C and D. It shows that in study D children have greater coverage with the Scottish curriculum. In both studies involving primary teachers they naturally use a create-first approach. Future work needs to plan and implement a study over a longer period of time in a typical classroom setting exploring the benefits of each method. Interestingly in the studies where children are asked what they want to learn about in computing science, most want to know about how computers work. The primary teachers in quadrant D demonstrated approaches to incorporate this into planned learning.

Finally, this thesis makes a contribution in providing insights into how primary teachers and computing science education can work collaboratively to improve outcomes for children.

10.1.1 Conclusion

This thesis explores the introduction of CS in Scottish primary schools, although arguably, it has implications for global practice. As far as the author is aware, at the time of writing, it is one of the few authentic primary school classroom introductory programming studies observing non-specialist primary teachers planning and delivering learning.

The intervention takes account of the comprehension-first approach research from HEI and in collaboration with participants, applies this to the primary teachers routines and instructional designs best suited for the particular needs of children in their classroom setting.

Contributions

This section considers the contribution to knowledge on introductory programming in primary school. Each RQ will be discussed in turn with the central point of the contribution being the create-first approach to the comprehension-first.

10.2 RQ1:

Is there value in computing science for all?

Quadrant A (2015) contributes to RQ1. It explores the voice of digital industries dominating the partnership working within CS curricula development for primary school amidst claims that the thinking skills of a programmer are of value to all irrespective of their chosen career. A study with non-digital industries shows similarities in a spread of non-digital industries views of effective employees skills. The study proposes that these skills align to process thinking within the CS programming domain.

This quadrant leads to the need to explore If computing science is being taught in primary then are learners developing those much-valued skills for all.

10.3 RQ2:

What typical programming approaches are in place in traditional primary classrooms and how successful are they?

Quadrant B and quadrant D address RQ2. Quadrant B arises from quadrant A findings that CS education is of value to all and literature shows it is being taught in primary classrooms.

The quadrant starts with relevant literature on typical methods in primary school and finds that in most studies researchers use visual programming environments. However, few studies explore how primary teachers naturally plan and implement CS in a typical primary classroom setting. The fieldwork study (2016) took place in 4 different schools with 4 teachers. Observations show that all lessons are delivered through a create-first approach and a few children are demotivated by the experience. Interestingly, Quadrant D study also explores typical approaches in the primary classroom and the same create-first approach is observed. This is interesting because quadrant D study takes place two years later than quadrant B study and the CS curriculum promoting a comprehension-first approach is now in place.

In summary, the observations around research question RQ2 indicate that the typical approach to programming learning in primary schools by non-specialist teachers is to use a create-first approach using visual / block-based programming environments.

10.4 RQ3

Can a comprehension-first introductory programming course be implemented successfully in a primary school classroom?

Quadrant C (2017) and Quadrant D (2018) contribute to answering RQ3. In 2017, the CfE CS curriculum is refreshed aligning itself to a comprehension-first approach. Given the limited literature insights on a comprehension-first approach in primary, a study based on this approach was implemented using CS expertise in primary classroom settings. These experts had no primary school teacher training.

Findings from this study demonstrate that a comprehension-first approach can be delivered by educators without a primary background. However, each class was supported by the class teacher. Overall, children progressed well and in a short teaching time covered almost half of the 1st level CfE CS curriculum. In addition, children's reflections changed from describing the activity in the unplugged sessions to including more technical vocabulary. This change implied that children.

Quadrant D also contributes to RQ3 and was carried out in 2018. By this time, there is more literature with insights into comprehension-first primary school programming approaches. However, studies continue to control conditions and would arguably be challenging to upscale and replicate in a typical classroom setting. The study in quadrant D is in two phases, recognises the challenge of curriculum adoption. Firstly, primary teachers who have not received explicit CS training plan and implement a course using materials they discover on-line. As mentioned above, the primary teachers employ a create first approach. In phase two, the teachers modify their planning and implementation applying the comprehension-first approach. The table below captures the approaches in phase 1 and phase two. Phase two shows that during the same duration as phase one, teachers embed the comprehension-oriented approach within their typical practice. This approach enables children to gain broader coverage of the CfE curricula expectations. In addition, in phase two, in phase two children experience a high quality literacy environment within the IP learning.

Phase 1	CfE 3 - steps	Phase 2	CfE 3-step
Tasks children undertake in a 3-week period		Tasks children undertake in a 3-week period	
Complete a partial program in order to achieve an output and reinforce a concept. (Independently)	Step 3	Describe the process observed in the real/virtual world. (Using think, pair share strategies)	Step 1
Modify segments of code to complete a program with a given goal. (Independently)	Step 3	Connect blocks to the processes observed. (Using think, pair share strategies)	Step 2
Create a program from pre-defined blocks when the goal of the program is given.	Step 3	Predict the behaviour of a given program. (Using think, pair share strategies)	Step 2
Predict the behaviour of a given program <i>(School K only)</i>	Step 2	Modify a program in order to achieve a given output. (Using think, pair share strategies)	Step 3
		Complete a partial program in order to achieve a given output	Step 3
		Create and describe programs accurately using computational tools. (Using think, pair share strategies)	Step 3

Table 75 Phase one and phase two tasks in relation to CfE 3 step approach

Conclusion

In quadrant C, the implementation of the comprehension-first approach was successful with high levels of motivation and children demonstrating an understanding of code before

they created scripts. In quadrant D, working closely with primary teachers resulted in their application of typical classroom management and organisation strategies to the IP. These strategies provide children with purposeful and manageable opportunities for listening and talking about the scripts, their goals, functions and assign meaning to the blocks.

Using block base programming environments involves children reading. At the end of the study, one of the teachers aligned the create-first approach with the teaching of writing to children in literacy, something that every primary teacher will know about already. She commented that teaching of writing relies heavily on high quality listening and talking, being read to, before reading before writing. This is a model that could very valuably be used when explaining the approach to primary teachers more broadly.

The connection between a comprehension-first approach to programming and to typical natural language literacy approaches already used at the primary level is a major realisation and contribution of the thesis, enabling primary teachers' existing knowledge to be built upon and significantly simplifying the transition for these teachers as they are asked to include this new curriculum area in their portfolio.

Personal reflections

The six year duration of this part-time PhD has led to some very significant realisations that run alongside the thesis statement and research questions and relate to my professional work that has continued alongside these studies. In particular, the challenges of balancing education policy implementation alongside accessible and high quality professional learning, the need for collaborative approaches with primary teachers' and the CS education community to learn and develop with each other as equal partners, and the safe intellectual space of the CS education community and the work that they undertake. In my professional work, I am more informed about challenging my own and others' assumptions about national advice on policy decisions and aware of how our practice of evidence gathering and dissemination for 'grey' literature differs from rigorous academic studies. However, for me, the strongest outcome is finally being able to evidence and articulate the link between teaching literacy to young children and the teaching of programming languages in primary school. Both rely heavily on effective listening and talking strategies to develop children's understanding through observing the processes all around them.

Chapter 11

Epilogue

11.1 Background

A central theme in the thesis is the exploration of primary school teacher's delivery of introductory programming in formal education during 2016 to 2019. Chapter 9 considers contributions from literature to the thesis contents from 2019 until 2022.

11.2 Introduction

The chapter is set within the 2019-2022 timeframe. It aims to provide an overview of introductory programming within the primary school context. Reading shows introductory programming in the primary school curriculum is often, rightly or wrongly, used interchangeably with computational thinking (CT). While efforts continue to be made to define CT, a consensus remains open to interpretation. In fact, a number of papers use the term computing science, computing, or CT when framing key concepts and practices in programming. Fagerlund, Hakkinen et al., (2021) J., Hermans, F., (2016) for example, describes CT as an elusive term and programming centric. Defining CT and its relationship with CS is touched on lightly within the thesis and outwith the scope of this chapter. However, due to the interchangeable terminology used within primary school introductory programming studies, term CS/CT will be used when referring to computing science curricula. An approach adopted by Salac, J., Butler., et al., (2021) that enables inclusion of a broader range of studies.

Literature shows during the updated timeframe of 2019 -2022 from the original thesis, that CS/CT is expanding globally. Curriculum content remains broadly the same with acknowledgment that advancements in technology may impact on curriculum reviews. Insights continue to focus children's experiences using adaptations of HEI introductory programming interventions. Importantly, Scratch examples now incorporate program comprehension. In these examples, children now have the opportunity to use and modify scripts before creating their own. This advancement is worth noting because Scratch was originally developed promoting only a create first approach.

This aim of this chapter is to explore and updates the central themes of the thesis.

- *the expansion of CS in primary school curriculums*
- *the justification of CS in the primary school curriculum*
- *the primary school CS curriculum content. and,*
- *primary school pedagogy*

In line with the writing conventions throughout the thesis, the terms CS/CT and IP are used interchangeably.

11.3 Computing Science Curriculum is expanding globally

Increasingly, countries continue to introduce CS/CT at the primary stages of a child's compulsory education (Salac, Thomas, et al., 2021) (Tsarava, Moeller, et al 2022) (Salac, J., 2020). It is noted that the pace of implementation and content varies with no accurate global overview. Fagerlund, J., (2021), also highlights that while many countries deliver CS/CT few incorporate programming in primary education as a compulsory topic. This view is supported by Kober who reports that programming is not being taught in Austria or Germany primary schools (Körber, Bailey, et al., 2021). While there is some attempt to gather this information through the work of Falkener, K and Quile. K., et al (2019), Tsarava, K., Moeller, K., et al (2022) state that helpful review is needed across pre-school, primary, secondary and University. They suggest that with the expansion of the CS curricula there is a need for a better understanding and further insights into CS pedagogy in primary schools.

11.4 Curriculum justifications.

The relevance of a computing science curricula seems considerable, because of the highly computerised world that we live in (Tsarava and Moeller, 2022). Arguably, the problem solving skills of a computer programme developed through programming are much needed and transferable. Wing's (2008) view that CT is a skill essential for everyone and not only programmers or computer scientists is well known. She claims that CS/CT is a fundamental and important skillset in our world and should be taught and practiced in school and therefore a necessary component of general education (Arnold, Amstalden, and Bader, 2022). There is potential that CS skills are applicable in a range of problem-solving situations (Fagerlund, Häkkinen, et al., 2021).

Other justifications for CS curriculum in formal primary school education include its contribution to improving the lack of STEM professionals and teachers of STEM in the system alongside cognitive benefits (Yeni, Grgurina, et al., 2021). It is suggested that (CS/CT) is one of the main contributions to general education because it is a mental discipline for thinking about doing computation of all kinds (Arnold, Amstalden, et al., 2012). With this claim in mind, (Scherer, Siddiq, 2022) study on 192 third and four graders found positive associations with a CT assessment and complex numerical abilities, verbal reasoning abilities and non-verbal visuospatial abilities.

Salac's (2021) work explores the benefits of CS/CT but recognises the need to for researchers and educators to support claims with reliable evidence. Her work focuses on demonstrating impact on the core academic outcomes such as reading and math that, she believes, are most valued by schools. In her quasi-experimental study on children age 9-10 she compared reading and maths attainment on a state wide test of those receiving CS/CT instruction. Findings show less scaffolding and more open ended CS/CT instruction was associated with performance gains in math, for children with low proficiency in English, from areas of social economic deprivation and but not for children with additional support needs. While findings are compelling for CS/CT inclusion, Salac conclude that there is still a gap in understanding computing and its relationships with other subjects.

11.5 Primary School computing science (CS/CT) curriculum content

Although this chapter is focused on the computing science curriculum as a whole, it is recognised that in primary school curriculums, computational thinking is an important aspect taught primarily through programming (Körber, Bailey, et al., 2021) (Scherer, R., Siddiq, 2020). Terminology continues to vary with Fagerlund, J., Häkkinen, P., et al., (2021) highlighting in some literature there can be no reference to CS when discussing CT and similarly no mention of CT when describing CS. He believes that this could be attributed to the continued debate on defining and operationalising CT, he does state that there is some consensus that CT is developed in part through programming. Fagerlund, J., Häkkinen, P., et al., (2021) in his systematic review challenge fixed frameworks. For example, Brennan, K. and Resnick, M., (2012) CT Framework is that context specific frameworks therefore may be unsuitable for framing CT across programming contexts and promoting deeper learning. He suggests that a CS goal is for children to learn:

- what computers can and cannot do

- understand how computers do the things they do
- apply computational tools, models and ideas to solve problems in various contexts.

This three step approach broadly reflects the Scottish curriculum whereby children

- Understand computation
- Understand the tools for the computation
- Apply the tools and build solutions

CS curricula content detail and starting age varies across countries. Scotland, for example, provides three broad organisers for all children in formal education age 3 to 15. Teachers personalise content suitable to relevant contexts, the age and stage of children’s development building on prior learning. New Zealand’s CS/CT offer starts at age 5 with a detailed programme of study. Given the technological advancements and pace of change, with for example AI, the CS/CT primary school curricula may need constant review to ensure that it is fit for purpose.

In terms of CS/CT curriculum content is delivered either as a discrete stand alone subject and/or through interdisciplinary learning (IDL) where it is taught through other subjects (Arnold, Amstalden, et al., 2022). In addition, there is increased attention of CT skills in many subjects stating that although CT is often regarded as related to computer science (CS) and relevant for mathematics and scientific contexts, CT is concerned with problem solving in every domain and may be instrumental in every possible field. Therefore, teaching CT in primary should not only take place in the context of CS or STEM subjects but can be very well embedded in language classes, arts and humanities. However, the findings are based on 8th grade students and although the paper is within the broader K-12 learning trajectories, it is not clear if their approach could be replicated with very young children (Yeni, Grgurina, et al., 2021).

It can be suggested that CS/CT will continue to develop. ‘People’s lives today are full of ML driven services and the curriculum needs updated to reflect changes in society (Tedre, 2022).

“Programming is being widely adopted as a classroom activity to promote computational literacy across the full spectrum of ages. As of now, however, there is a gap between curriculum designers and the teachers that work directly alongside pupils. Educators build

their lessons around predefined curricula and programming environments with limited scope for customization (Staub, Chothia, et al., (2021).”

11.6 Computing science pedagogy at primary school level.

With the expansion of a CS/CT curriculum in formal education, more children at an earlier age of formal education access CS/ CT through programming (Velázquez-Iturbide, Paredes-Barragán, et al., 2022) (Körber, Bailey, et al., 2021). There continues to be an increasing focus on gaining pedagogical insights that achieve success for all children (Salac, 2020) . However, almost a decade after Wing’s seminal statement, the best methods for teaching programming to young children are still to being explored (Körber, Bailey, et al., 2021). Interventions focus on adapting successful HEI interventions such as program comprehension. Child development theorists may question the appropriateness of adult learning adaptations and importantly, there is an unintended consequence that the challenges found in HEI will be replicated for younger children through these adaptations.

‘It is critical for computer science instruction to be effective for a broad spectrum of students. Diverse learners significantly underperform white peers with higher socio-economic status on important academic markers (Franklin, Salac, et al., 2020).

However, despite the scepticism that young children are very different from adult learners, it is worth noting that a study of 192 third and fourth grade children’s CT assessments and cognitive functions aligned with students in secondary and university results (Tedre, 2022).

11.7 Teacher’s PCK

There is a recognition that alongside the challenge of defining a curriculum there is a need to ensure that sufficient training is in place to train teachers to deliver it. Teacher development is highly important for any subject including CS/CT/IP education of young children. In introductory programming teacher’s need to understand both the tools for instruction alongside the concepts that are being taught. Lee Shulman’s Pedagogical Content Knowledge (PCK) (1986a) argues that teachers need, subject knowledge and general pedagogical skills, in order to teach topics in ways that learners can understand. Teacher’s PCK is representing and formulating the subject that makes it comprehensible to other.

Exploration of curriculum CS/CT PCK for primary teachers delivering introductory programming for young children in formal education is increasing. One framework (Yeni, Grgurina, et al., 2021) operationalises PCK into themes that describe the need to understand both what teachers need to know and what they are able to do. The framework considers and approach to teach a new subject. They appreciate that there is a challenge how to integrate it. The themes highlight instructional strategies as important alongside bridging the gap between teacher's existing knowledge and that of the new subject. PRIMM, TIPP and SEE sit well within the instruction whereas STEP (Cole, 2022) can be shown to bridge the gap between the experienced primary teachers application of a range of subjects to CS. A case study set out to illustrate a framework to guide and design the delivery of lessons. Researchers asked primary and secondary schools teachers views on what's needed to teach the subject returned interesting results. Teachers said that they need to know about CS/CT integration, attitudes and barriers. Authors conclude that teachers also need to be more capable of technical knowledge. Interestingly in the study there is no mention of program comprehension and the role that it plays.

11.8 Instructional methods

Research tells us that a common approach to teaching programming concepts is through an unplugged approach. Unplugged activities are defined as CS/CT activities that require no computers (Körber, Bailey, et al., 2021). This approach usually involves an element of fun and motivates children. Unplugged, does present challenges with transferring the unplugged activities to computers. This issue led to research focused on improving understanding of how to bridge the unplugged approach with computers. In their study in 8 primary schools in Germany and Austria, authors propose that physical computing with programmable devices such as robots is a possible intermediary step. This is because they are tangible and provide real world impact through their actions. Interestingly, this study aims to support the transfer from unplugged to plugged taking account of cognitive load theory and program comprehension through use, modify create. This focus is a positive step of bringing together three well documented approaches in HEI introductory programming to solve a specific gap in knowledge for younger children.

Alongside curriculum content a number of programming tools are available to support the introduction of programming. However, Scherer, R., Siddiq, F. et al., (2020) posits superiority over some programming tools than others is attributed to their visual nature

which makes programming more accessible. They state that as early as the 1990s customising programming tools and languages for certain age groups of students especially younger students is considered an integral part of programming instructions. Griefston also supports this view commenting that block based programming is often used as an entry point to programming since the block based nature reduces the complexity of text based programming languages (Greifenstein, Obermüller, et al., 2021). Fagerlund claims that that programming with Scratch the graphical based programming tool is especially popular with the younger age group because of the low flow required. However, he does highlight that ‘despite the affordance of graphical tools, programming is cognitively complex’.

Professional learning opportunities for teachers are available whereby they learn about teaching CS concepts through the use of approaches such as code.org or unplugged activities. However, there is little evidence showing teacher’s PCK development distinguishes between:

- developing teacher’s understanding of CS concepts from
- developing teacher’s understanding of using the tools for instruction.

This is important because the two components are very different although undoubtedly However, if the teachers don’t understand the concepts underpinning the blocks then neither will the children they are teaching.

Scratch Jr is an example of where misconceptions may arise. This visual programming environment was developed using a learner-centred approach. It requires teachers to develop as part of their content knowledge an understanding of the both the language and programming basis. However, it is possible that when learning to use the Scratch Jr, teachers focus on the programming tools and not the CS/CT concepts

While a number of authors reference block programming more broadly as a programming tool. Franklin specifies that CS/CT curriculums in primary school is often delivered using Scratch, a block programming language (Franklin, Salac, et al., 2020). As mentioned previously, this brings a challenge because Scratch programming tools was not envisaged for teacher development but for teacher instruction, therefore, the potential issue is that teachers can use Scratch but they may not have a deep understanding of block based programming (Franklin, Salac, et al., 2020).

In addition, there is a view that the block programming environment while addressing some of the programming complexities, it, may not be developmentally appropriate for early readers or early numeracy because the environment uses the Cartesian coordinate

system for character movement (Velázquez-Iturbide, Paredes-Barragán, et al.,2022). Moreover, and this thesis focus, Scratch lends itself to build first approach without considering comprehension first.

Franklin, and Salac, (2020) explore Scratch charades an intervention designed to allow children to learn Scratch programming language outside of the Scratch development environment. Scratch charades uses demonstration through gamification. Their study uses Scratch Charades to gain insights into early interpretation of Scratch blocks and scripts. The study shows misconceptions in children's interpretation of blocks. They found that while students understand the concept of sequential and loop-based code very easily, a few commonly-used Scratch blocks can lead to confusion.

11.9 CS/CT pedagogical links with literacy

The role of language in the field of computer science education is seen as an important aspect that is often overlooked when designing and implementing introductory programming courses. Learning to program is often compared to learning a natural language, one difference being, you need precise language for programming because you are communicating with a machine instead of another human. This programming ability and language link is also demonstrated in studies involving younger children. Tsarava, K., Moeller, K., et al., (2022) study on 28, 3 to 6 year olds found a positive association between cognitive compiling of syntax in natural language and programming ability; Howland, K., (2015) found the development of CT through programming is enhanced by scaffolding activities that switch between formal and natural language; and, make claims that language is relevant for programming and CT citing several empirical studies showing the correlation.

See Talk Explain and Prepare (STEP) (Cole, E, Cutts, Q 2022), PRIMM (Sentence, S., 2017) and TIPP and SEE (Salac, J., 2020) interventions acknowledge the important role of natural language when learning to program. These scaffolding techniques sit within the program comprehension approach whereby children understand the visual programming blocks at different levels before 'jumping' straight into writing or 'creating' a script that they may not understand. These interventions support the view that students who had better skills at reading, tracing and explaining code tended to be better at writing code.

Program comprehension is recognised as an important step in learning to program. However, when teaching a young child to read and write in their natural language, listening and talking is another important step that needs considered. Parry's (2020) action research experimental study explored programming taking account of the process for children learning to read, speaking and writing in their natural language. Additional language exercises using stratagrams as the main intervention on 60 students age 12-13 made a positive impact on student's learning.

Salac, J., (2020) bases her work in Schulte's block model but clarifies program comprehension within the context of young children at primary school level. She explains that students need to comprehend the meaning of instructions, similar to how they need to comprehend the meaning of words when reading. In reading varied text structures, conceptual density and technical vocabulary require readers to recognize patterns, develop mental models of abstract vocabulary concepts, and understanding the technical meaning of disciplinary vocabulary. Programming likewise, requires comprehension of varied text structures, technical vocabulary, pattern recognition and mental models of abstract concepts. Like reading, programs have text structures which convey meaning and have purpose. They also comment on CS/CT as a single word which is helpful as the CS/CT relationship has not yet been established (Salac, and Thomas, 2021).

At the original time of writing the thesis 2016 -2019, PRIMM was mainly used in text based languages at secondary school and upper primary builds. In 2022⁸, Scratch has incorporated prepared projects using PRIMM for children and teachers to use-modify and create. Although at this stage, small in numbers, this is an important advancement because it challenges the original Scratch approach that promoted children to build scripts through exploration. This progress enables further research for the community interested in providing insights into the create first verses comprehension first approaches. Indeed, even now, it does not feature in many of the literatures in primary school contexts and it can be considered that this area remains relatively under explored (Parry, A., 2020). STEP focuses on discussions on the overall goal from Schulte (2008).

⁸ <https://scratch.mit.edu/studios/27510835>

11.10 CS/CT links to other subjects

CS/CT is often linked with skills in maths often looking at predictors of success in HEI introductory programming courses. Salac's (2020) study explores the reverse relationship of CS/CT in primary school with maths and reading. The study uses the TIPP & SEE learning strategy scaffolding students' exploration of example Scratch programs. Findings claims that CS/CT improves outcomes in maths for children living in socio-economic deprivation and those with EAL. Salac, J., (2020) rightly concludes that Scratch covers some advanced maths concepts ordinarily too advanced for the age/grade levels. Therefore, using Scratch may contribute to the impact on maths capabilities (Salac, Thomas, et al., 2021)

11.11 Conclusion

The purpose of this chapter is a view of computing science in primary school formal education from a different period of time (2019-2022) of the thesis body. It is clear that the expansion on CS continues with growing attention for CS/CT integration. Primary school teachers continue to use visual programming environments such as Scratch to deliver content. While the intended curricula content may be age and stage appropriate, it's enactment also relies heavily on appropriate planning of visual programming environments as a tool for delivery.

There is an increasing interest in investigating teachers' pedagogical content knowledge. However, typical pedagogy employs much researched university program comprehension approaches adapted for younger children through scaffolding techniques such as PRIMM, USE modify Create and TIPP and SEE continue to be developed and improved. Literature continues to be limited on considering if a create-first or comprehension-first approach when analysing interventions. In addition, it is worth noting that, Scratch was developed for teacher instruction and not teacher development (Velázquez-Iturbide, Paredes-Barragán, 2022). By omitting teachers language and programming basics, it could create misconceptions. Teachers need a deep understanding of block programming as well as the concepts.

Primary teachers are often referred to as 'lacking in confidence' or unskilled and a positive move is referring to teacher training needs (Fagerlund, Häkkinen, 2021). This simple use of terminology promotes an improvement on the typical deficit labelling that teachers do not have the skills to teach CS/CT to younger children. With that in mind, it is important

to recognise that Scratch was created for teacher instruction and not teacher development. Therefore teacher training should consider the CS/CT concepts being taught as distinct from developing teachers skills in using this programming tool. Within the comprehension approach 3 years on and it's still PRIMM, Use modify Create, TIPP and SEE, now seeing diagramming . While PRIMM UMC TIPP and SEE are such examples of improving teacher's use of the programming tools by providing mnemonics for children STEP (Cole, 2022) adds to these approaches addressing skilled primary school teachers PCK transfer by linking their introductory programming approaches to their teaching of reading signalling the importance of listening and talking. Of note is Scratch incorporating a few UMC examples in their library.

Finally, given the increase in programming within formal school curriculums, it is likely that the research interest will continue to grow. There are encouraging steps towards literature on programming strategies that promote the use of language as well as teaching tools that have been successfully adopted from natural language education. However, this chapter concludes that there is scope to further investigate the connections between these domains (Parry, 2020). In particular there is a need to develop primary teacher's understanding of the key concepts as well as their use of block based programming as an instruction. Otherwise, there is a risk of replicating the challenges of introductory programming in HEI on all children undertaking a CS/CT curriculum.

'Given the increasing demand for programmers, the move of programming into school curriculums, and the well documented challenges involved, the topics of teaching and learning programming are likely to remain of significant interest in computing education for the foreseeable future.'

References

- 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. *Journal of Educational Technology & Society*, 19(3), pp.47-57.
- Armoni, M., 2012. Teaching CS in kindergarten: How early can the pipeline begin?. *Acm Inroads*, 3(4), pp.18-19.
- Arnold, R., Amstalden, B. and Bader, J., 2022, October. Enhancing the Role of Computational Thinking in Primary and Secondary Education in Switzerland. In *Proceedings of the 17th Workshop in Primary and Secondary Computing Education* (pp. 1-2).
- Asia News 2016. Teaching the next generation to express themselves in code. Available at <https://news.microsoft.com/apac/features/teaching-chinas-next-generation-to-express-themselves-in-code/>. Accessed 24 December 2021
- Athanasiou, L., Topali, P. and Mikropoulos, T.A., 2016, November. The use of robotics in introductory programming for elementary students. In *International Conference EduRobotics 2016* (pp. 183-192). Springer, Cham
- Australian Curriculum Assessment and Reporting Authority (ACARA) (2014). *Australian Curriculum: Digital Technologies*.
- Bansal, A., Aggarwal, P. and Singh, M., 2020. Employability of engineering students-a report on the discernment of the students and their employers.
- Bargury, Iris & Muller, Orna & Haberman, Bruria & Zohar, Doron & Cohen, Avi & Levy, Dalit & Hotoveli, Reuven. (2012). Implementing a new Computer Science Curriculum for middle school in Israel. *Proceedings - Frontiers in Education Conference*.
- Benitti, F.B.V., 2012. Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, 58(3), pp.978-988.
- Bergin, S. and Reilly, R., 2006. Predicting introductory programming performance: A multi-institutional multivariate study. *Computer Science Education*, 16(4), pp.303-323.
- Bers, Flannery, L., Kazakoff, E.R. and Sullivan, A., 2014. Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, 72, pp.145-157.
- Bickman, L. and Rog, D.J. eds., 2008. *The SAGE handbook of applied social research methods*. Sage publications.

- Biesta, G., 2007. Bridging the gap between educational research and educational practice: The need for critical distance.
- Black, P. and Wiliam, D., 2010. Inside the black box: Raising standards through classroom assessment. *Phi delta kappan*, 92(1), pp.81-90.
- Bocconi, S., Chiocciariello, A. and Earp, J. (2018). The Nordic approach to introducing Computational Thinking and programming in compulsory education. Report prepared for the Nordic@BETT2018 Steering Group.
<https://doi.org/10.17471/54007>
- Braun, V. and Clarke, V., 2006. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2), pp.77-101.
- Brennan, K. and Resnick, M., 2012, April. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada* (Vol. 1, p. 25).
- Brendan Coates, Matt Cowgill, Tony Chen, and Will Mackey. 2020. Shutdown: estimating the COVID-19 employment shock. Grattan Institute, Melbourne, Victoria (2020).
- Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? *ACM Inroads* 2, 1 (March 2011), 48-54.
- Brioso, X., 2015. Integrating ISO 21500 guidance on project management, lean construction and PMBOK. *Procedia Engineering*, 123, pp.76-84.
- British Telecom 2016 Tech Literacy A new cornerstone of modern primary school education A report from BT and Ipsos MORI
- Brooks-Gunn J, & Duncan GJ (1997). The effects of poverty on children. *The Future of Children*, 7, 55–71. 10.2307/1602387
- Brosnan, M., 1998 The role of psychological gender in the computer-related attitudes and attainment of primary school children (ages 6-11) *Computers and education* 1998
- Brophy, J., 2006. Observational Research on Generic Aspects of Classroom Teaching.
- Bush, L., 2003 Preparing Children To Read and Learn: An Education Initiative of Laura Bush. ERIC
- Butler D, Leahy M. Developing preservice teachers' understanding of computational thinking: A constructionist approach. *British Journal of Educational Technology* 2021 05;52(3):1060-1077
- Cansu Atlay, Nicole Tieben, Steffen Hillmert, and Benjamin Fauth. 2019. Instructional quality and achievement inequality: How effective is teaching in closing the social achievement gap? *Learning and Instruction* 63 (2019), 101211.
- Charlotte Hill†, Hilary A. Dwyer‡, Tim Martinez†, Danielle Harlow‡, Diana Franklin†

- Charmaz, K., 2006. Constructing grounded theory: A practical guide through qualitative analysis. sage.
- Charoula Angeli, et al. "A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge." *Journal of Educational Technology & Society*, vol. 19, no. 3, International Forum of Educational Technology & Society, 2016, pp. 47-57, <http://www.jstor.org/stable/jeductechsoci.19.3.47>.
- Chris Stephenson, Steven Cooper, Barbara Boucher Owens and Judit GalEzer. 2012. The new CSTA K-12 computer science standards. In Proceedings of the 17th ACM annual conference on innovation and technology in computer science education. 363-364
- Code.org <https://code.org/educate/curriculum/csf> Accessed 27/02/23
- Cohen J (1960) A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20: 37-46
- Comber, C., 1997. et al The effects of age, gender and computer experience upon computer attitudes *Educational Research* (1997)
- Communications of the ACM*. 62. 54-55. 10.1145/3343445.
- Cordingley, P., Bell, M., Thomason, S. and Firth, A., 2005. The impact of collaborative continuing professional development (CPD) on classroom teaching and learning. Review: How do collaborative and sustained CPD and sustained but not collaborative CPD affect teaching and learning.
- Curriculum Advisory Group Report DIGITAL TECHNOLOGIES & HANGARAU MATIHIKO OCTOBER 2017
<https://www.education.govt.nz/assets/Documents/dthm/Nov-17-update/Digital-Curriculum-Consultation-CAG-Report.pdf> .
- Curtis, D. and McKenzie, P., 2001. Employability skills for Australian industry: Literature review and framework development. Melbourne: Australian Council for Educational Research.
- Cutts, Q., Patitsas, E., Cole, E., Donaldson, P., Alshaigy, B., Gutica, M., Hellas, A., Larraza-Mendiluze, E., McCartney, R. and Riedesel, C., 2018, January. Early developmental activities and computing proficiency. In Proceedings of the 2017 ITiCSE Conference on Working Group Reports (pp. 140-157).
- Cutts, Q., Robertson, J., Donaldson, P. and O'Donnell, L., 2017. An evaluation of a professional learning network for computer science teachers. *Computer Science Education*, 27(1), pp.30-53.
- Dambre, J. et al. (2012) 'Information processing capacity of dynamical systems', *Scientific Reports*. doi: 10.1038/srep00514.

- Daniels, D.H., Kalkman, D.L. and McCombs, B.L., 2001. Young children's perspectives on learning and teacher practices in different classroom contexts: Implications for motivation. *Early Education and Development*, 12(2), pp.253-273.
- Day et al, 2007), Day, C., P. Sammons, G. Stobart, A. Kington, and Q. Gu. 2007. *Teachers Matter: Connecting Work, Lives and Effectiveness*. London: Open University Press.
- Denning, P. J. (2009) 'The profession of IT: Beyond computational thinking', *Communications of the ACM*. doi: 10.1145/1516046.1516054.
- Denning, P.J., 2005. Is computer science science?. *Communications of the ACM*, 48(4), pp.27-31.
- Developing and implementing a new computing science curriculum in Scottish schools | The University of Edinburgh <https://www.ed.ac.uk/education/rke/making-a-difference/new-computing-science-curriculum> accessed 22/12/2021
- Digital Technologies Hub. Available at <https://www.digitaltechnologieshub.edu.au/> Accessed 24 December 2021
- Dziallas, S. and Fincher, S., 2019, July. Accountable disciplinary knowledge in computing education: A case-comparative approach. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (pp. 1-9).
- Edmonds, W. and Kennedy, T. 2017. Convergent-Parallel Approach. In: *An Applied Guide to Research Designs: Quantitative, Qualitative, and Mixed Methods*, Second ed. Thousand Oaks, CA: SAGE Publications, Inc pp. 181-188. Available at: <<http://www.doi.org/10.4135/9781071802779>> [Accessed 1 Jan 2022]
- Education Scotland Technologies experiences and outcomes (2017). Available at <https://education.gov.scot/Documents/Technologies-es-os.pdf> Accessed 24 December 2021
- Education Scotland Technologies: Principles and practice (education.gov.scot). Available at <https://education.gov.scot/Documents/technologies-pp.pdf>. Accessed 24 December 2021
- Efthimia Aivaloglou and Felienne Hermans. 2016. How kids code and how we know: An exploratory study on the Scratch repository. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. 53-61.
- ET, O.D.C., 21st Century Learning: Research, Innovation and Poli
- Fagerlund, J., Häkkinen, P., Vesisenaho, M. and Viiri, J., 2021. Computational thinking in programming with scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, 29(1), pp.12-28.

- Falkner, K. et al. (2019) 'An international comparison of K-12 computer science education intended and enacted curricula', in ACM International Conference Proceeding Series. doi: 10.1145/3364510.3364517.
- Farrel, Robertson, Cutts and Connor (2017). Teach computing science a guide for primary schools and early years practitioners 2017 (updated 2018) Available at <https://teachcs.scot>. [Accessed 24 December 2021]
- Fatimah, Y.A., Govindan, K., Murniningsih, R. and Setiawan, A., 2020. Industry 4.0 based sustainable circular economy approach for smart waste management system to achieve sustainable development goals: A case study of Indonesia. *Journal of Cleaner Production*, 269, p.122263.
- Fatourou, E., Zygouris, N.C., Loukopoulos, T. and Stamoulis, G.I., 2021, April. Review of Learning Design Choices of Primary School Programming Courses in Empirical Researches. In 2021 IEEE Global Engineering Education Conference (EDUCON) (pp. 1010-1018). IEEE.
- Fereday, J. and Muir-Cochrane, E., 2006. Demonstrating rigor using thematic analysis: A hybrid approach of inductive and deductive coding and theme development. *International journal of qualitative methods*, 5(1), pp.80-92.
- Ferrara, K., Hirsh-Pasek, K., Newcombe, N.S., Golinkoff, R.M. and Lam, W.S., 2011. Block talk: Spatial language during block play. *Mind, Brain, and Education*, 5(3), pp.143-151.
- Fielding, M., Bragg, S., Craig, J., Cunningham, I.A.N., Eraut, M., Gillinson, S., Horne, M., Robinson, C. and Thorp, J., 2005. Factors influencing the transfer of good practice. *Floors and Flexibility: Designing a Programming Environment for 4th-6th Grade Classrooms* Charlotte Hill[†], Hilary A. Dwyer[‡], Tim Martinez[†], Danielle Harlow[‡], Diana Franklin[†]
- Florian R Hertel and Nadine M Schöneck. 2019. Conflict perceptions across 27 OECD countries: The roles of socioeconomic inequality and collective stratification beliefs. *Acta Sociologica* (2019), 0001699319847515.
- Fluck, A. et al. (2016) 'Arguing for computer science in the school curriculum', *Educational Technology and Society*.
- Francisco J Gutierrez, Jocelyn Simmonds, Nancy Hitschfeld, Cecilia Casanova, Cecilia Sotomayor, and Vanessa Peña-Araya. 2018. Assessing software development skills among K-6 learners in a project-based workshop with scratch. In 2018
- Franklin, D., Salac, J., Thomas, C., Sekou, Z. and Krause, S., 2020, February. Eliciting student scratch script understandings via scratch charades. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 780-786)

- Funke, A., Geldreich, K. and Hubwieser, P., 2017, April. Analysis of scratch projects of an introductory programming course for primary school students. In 2017 IEEE global engineering education conference (EDUCON) (pp. 1229-1236). IEEE.
- Gardner, J. ed., 2012. Assessment and learning. Sage.
- Geetha Marcus. 2016. Closing the attainment gap: What can schools do. SPICe Briefing (2016).
- General Teaching Council for Scotland – Professional Standards for Teachers available at <https://www.gtcs.org.uk/professional-standards/professional-standards-for-teachers/>
- Google Education Digital Technologies Hub - website accessed <https://www.digitaltechnologieshub.edu.au/search#/site-search?pageNumber=1&keyword=GoogleCT>
- Greenwood, R., 2019. Pupil involvement in planning topics using KWL grids: opinions of teachers, student teachers and pupils. *Educational Studies*, 45(4), pp.497-519.
- Greifenstein, L., Obermüller, F., Wasmeier, E., Heuer, U. and Fraser, G., 2021, October. Effects of Hints on Debugging Scratch Programs: An Empirical Study with Primary School Teachers in Training. In *The 16th Workshop in Primary and Secondary Computing Education* (pp. 1-10).Paper 15
- Griffiths, D.A., Inman, M., Rojas, H. and Williams, K., 2018. Transitioning student identity and sense of place: future possibilities for assessment and development of student employability skills. *Studies in Higher Education*, 43(5), pp.891-913.
- Guzdial, M., Kay, A., Norris, C. and Soloway, E., 2019. Computational thinking should just be good thinking. *Communications of the ACM*, 62(11), pp.28-30.
- Harms, K.J., Balzuweit, E., Chen, J. and Kelleher, C., 2016, September. Learning programming from tutorials and code puzzles: Children's perceptions of value. In 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 59-67). IEEE.
- Hattie, J., 2012. Visible learning for teachers: Maximizing impact on learning. Routledge.
- Heintz, F., Mannila, L., Nygård, K., Parnes, P. and Regnell, B., 2015, September. Computing at school in Sweden-experiences from introducing computer science within existing subjects. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 118-130). Springer, Cham.
- Hijón-Neira, R., Pérez-Marin, D., Pizarro, C. and Connolly, C., 2020. The Effects of a Visual Execution Environment and Makey Makey on Primary School Children Learning Introductory Programming Concepts. *Ieee Access*, 8, pp.217800-217815.
- Hill, C., Dwyer, H.A., Martinez, T., Harlow, D. and Franklin, D., 2015, February. Floors and Flexibility: Designing a programming environment for 4th-6th grade

- classrooms. In *Proceedings of the 46th ACM technical Symposium on computer science education* (pp. 546-551).
- Hoffer., T., et al. 1995. Social Background Differences in High School Mathematics and Science Course taking and Achievement. *Statistics in Brief*. (1995).
- Holloway, I. and Brown, L., 2016. *Essentials of a qualitative doctorate*. Routledge.
- Howland, K. and Good, J., 2015. Learning to communicate computationally with Flip: A bi-modal programming language for game creation. *Computers & Education*, 80, pp.224-240.
- Hromkovi, J., Kohn, T., Komm, D. and Serafini, G., 2016, October. Combining the power of python with the simplicity of logo for a sustainable computer science education. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 155-166). Springer, Cham.
- Hutchinson, J., Dickinson, B., Vickers, R. and Hooley, T., 2015. The D2N2 employability framework: Employers and schools supporting young people's routes to work.
- Huth, M. and Ryan, M., 2004. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge university press.
- Ichinco, M. and Kelleher, C., 2015, October. Exploring novice programmer example use. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 63-71). IEEE.
- IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET). IEEE, 98-107.
- IPSOS Full report (bt.com)
- Kodu Game Lab | KoduGameLab Available at <http://www.kodugamelab.com/> accessed on 4 January 2022
- Iskrenovic-Momcilovic, O., 2018. Using computers in teaching in higher education. *Mediterranean Journal of Social Sciences*, 9(4), p.71.
- Israel. Proceedings - Frontiers in Education Conference. 1-6. 10.1109/FIE.2012.6462365.
- Jesús Moreno-León, Marcos Román-González, Casper Hartevelde, and Gregorio Robles. 2017. On the automatic assessment of computational thinking skills: A comparison with human experts. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2788-2795.
- John B Biggs and Kevin F Collis. *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press, 2014.
- José-Manuel Sáez-López, Marcos Román-González, Esteban Vázquez-Cano, *Journal of Technology and Science Education*. doi: 10.3926/jotse.271.

- Jun, S. J., Han, S. K. and Kim, S. H. (2017) 'Effect of design-based learning on improving computational thinking', *Behaviour and Information Technology*. doi: 10.1080/0144929X.2016.1188415.
- Karakhan, A., 2017. Six sigma & construction safety: Using the DMAIC cycle to improve incident investigations. *Professional Safety*, 62(6), p.38.
- Kashefpakdel, E. et al. (2019) 'How are schools developing real employability skills?', *Educationandemployers.Org*.
- Kathy A Mills. 2009. Floating on a sea of talk: Reading comprehension through speaking and listening. *The Reading Teacher* 63, 4 (2009), 325-329.
- Katrina Falkner, Rebecca Vivian, and Nickolas Falkner. 2014. The Australian digital technologies curriculum: challenge and opportunity. In *Proceedings of the Sixteenth Australasian Computing Education Conference - Volume 148 (ACE '14)*. Australian Computer Society, Inc., AUS, 3–12.
- Klein, L.L., Vieira, K.M., Feltrin, T.S., Pissutti, M. and Ercolani, L.D., 2022. The influence of lean management practices on process effectiveness: a quantitative study in a public institution. *SAGE Open*, 12(1), p.21582440221088837.
- Körber, N., Bailey, L., Greifenstein, L., Fraser, G., Sabitzer, B. and Rottenhofer, M., 2021, October. An Experience of Introducing Primary School Children to Programming using Ozobots (Practical Report). In *The 16th Workshop in Primary and Secondary Computing Education* (pp. 1-6).
- Kölling, M., 2015. Lessons from the design of three educational programming environments: Blue, BlueJ and Greenfoot. *International Journal of People-Oriented Programming (IJPOP)*, 4(1), pp.5-32.
- Kong, S.-C., Abelson, H. and Lai, M. (2019) 'Introduction to Computational Thinking Education', in *Computational Thinking Education*. doi: 10.1007/978-981-13-6528-7_1.
- Körber, N., Bailey, L., Greifenstein, L., Fraser, G., Sabitzer, B. and Rottenhofer, M., 2021, October. An Experience of Introducing Primary School Children to Programming using Ozobots (Practical Report). In *The 16th Workshop in Primary and Secondary Computing Education* (pp. 1-6).
- Laura B Perry and Andrew McConney. 2010. Does the SES of the school matter? An examination of socioeconomic status and student achievement using PISA 2003. *Teachers College Record* 112, 4 (2010), 1137-1162.

- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. and Werner, L., 2011. Computational thinking for youth in practice. *Acm Inroads*, 2(1), pp.32-37.
- LEGO® GB Available at <https://www.lego.com/en-gb> Accessed on 4 January 2022
- Lin, J.M.C., Lie, Y.L., Ho, R.G. and Li, C.C., 2007, October. Effects of guided collaboration on sixth graders' performance in logo programming. In 2007 37th Annual Frontiers In Education Conference-Global Engineering: Knowledge Without Borders, Opportunities Without Passports (pp. T1B-11). IEEE.
- Lincoln, Y.S. and Guba, E.G., 1985. *Naturalistic inquiry*. sage.
- Lopez, M., Whalley, J., Robbins, P. and Lister, R., 2008, September. Relationships between reading, tracing and writing skills in introductory programming. In Proceedings of the fourth international workshop on computing education research (pp. 101-112).
- Lu, J. J. and Fletcher, G. H. L. (2009) 'Thinking about computational thinking', *SIGCSE Bulletin Inroads*, 41(1), pp. 260-264. doi: 10.1145/1539024.1508959.
- Máiréad Dunne, Sara Humphreys, Judy Sebba, Alan Dyson, Frances Gallannaugh, and Daniel Muijs. 2007. Effective teaching and learning for pupils in low attaining groups. (2007).
- Margulieux, L.E., Morrison, B.B. & Decker, A. Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples.
- Marinus, E., Powell, Z., Thornton, R., McArthur, G. and Crain, S., 2018, August. Unravelling the cognition of coding in 3-to-6-year olds: The development of an assessment tool and the relation between coding ability and cognitive compiling of syntax in natural language. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*(pp. 133-141).
- Mayerová, K., 2012, April. Pilot activities: LEGO WeDo at primary school. In Proceedings of 3rd International Workshop Teaching Robotics, Teaching with Robotics: Integrating Robotics in School Curriculum (pp. 32-39).
- McLaren, S.V., 2011. Technologies in Scotland's Curriculum for Excellence: Time for Change. PATT 25: CRIPT8, p.285.
- McLachlan, C., 2018. Te Whariki revisited: How approaches to assessment can make valued learning visible.

- Medeiros, R.P., Ramalho, G. L. and Falcão, T. P. 2019 "A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education," in *IEEE Transactions on Education*, vol. 62, no. 2, pp. 77-90,
- Mensan, T., Osman, K. and Majid, N.A.A., 2020. Development and Validation of Unplugged Activity of Computational Thinking in Science Module to Integrate Computational Thinking in Primary Science Education. *Science Education International*, 31(2), pp.142-149.
- Michael J Lee and Amy J Ko. "Comparing the effectiveness of online learning approaches on CS1 learning outcomes". In: Proceedings of the eleventh annual international conference on international computing education research. ACM. 2015, pp. 237–246.
- Miles, M.B. and Huberman, A.M., 1994. *Qualitative data analysis: An expanded sourcebook*. sage.
- Mitchelmore, S. and Rowley, J., 2010. Entrepreneurial competencies: a literature review and development agenda. *International journal of entrepreneurial Behavior & Research*.
- Morgan, D.L., 2018. *Basic and advanced focus groups*. Sage Publications.
- Moschella, M. and Basso, D., 2020. Computational thinking, spatial and logical skills. An investigation at primary school. *Ricerche di Pedagogia e Didattica. Journal of Theories and Research in Education*, 15(2), pp.69-89
- Morrison, B.B., Margulieux, L.E., Ericson, B. and Guzdial, M., 2016, February. Subgoals help students solve Parsons problems. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (pp. 42-47).
- Mourshed, Chijioki & Barber, 2010). Mourshed, M., Chijioko, C. and Barber, M., 2011. How the worlds most improved school systems keep getting better. *Educational Studies*, (1), pp.7-25.
- Mourshed, M., Chijioko, C. and Barber, M., 2011. How the worlds most improved school systems keep getting better. *Educational Studies*, (1), pp.7-25.
- Natanson, M., 1968. Alfred Schutz on social reality and social science. *Social Research*, pp.217-244.
- National Research Council, 2010. Report of a workshop on the scope and nature of computational thinking. National Academies Press.
- Neale, J., Miller, P. and West, R., 2014. Reporting quantitative information in qualitative research: guidance for authors and reviewers.

- Neil CC Brown, Sue Sentance, Tom Crick, and Simon Humphreys. 2014. Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE)* 14, 2 (2014), 1-22
- New Zealand (Bell et al. 2012). At the same time some other countries, such as Israel, Poland and Cyprus, have maintained their particular Computer Science curricula and continued their development since the 1980s.
- Nowell, L.S., Norris, J.M., White, D.E. and Moules, N.J., 2017. Thematic analysis: Striving to meet the trustworthiness criteria. *International journal of qualitative methods*, 16(1), p.1609406917733847.
- OECD education policy implementation: A literature review and proposed framework
OECD working paper no. 162
- Panel: Extending and Evaluating the Use-Modify-Create Progression SIGCSE '20, March 11–14, 2020, Portland, OR, USA for Engaging Youth in Computational Thinking
- Parsons, D., & Haden, P. (2006). Parson's programming puzzles: A fun and effective learning tool for first programming courses. In D. Tolhurst & S. Mann (Eds.), *Computing Education 2006. Proceedings of the Eighth Australasian Computing Education Conference*, 52, (pp. 157-163). Adelaide, Australia: Australian Computing Society.
- Parry, A., 2020, October. Investigating the relationship between programming and natural languages within the PRIMM framework. In *Proceedings of the 15th Workshop on Primary and Secondary Computing Education* (pp. 1-10).
- Patacsil, F. F. and Tablatin, C. L. S. (2017) 'Exploring the importance of soft and hard skills as perceived by it internship students and industry: A gap analysis',
- Patton, M.Q., 1987. *How to use qualitative methods in evaluation* (No. 4). Sage.
- Paul Black, Christine Harrison, Clare Lee, Bethan Marshall, and Dylan Wiliam. 2004. Working inside the black box: Assessment for learning in the classroom. *Phi delta kappan* 86, 1 (2004), 8-21.
- Pavel Valentinovich Borbotko. 2019. DIGITAL SKILLS CRISIS AND REVISION OF EDUCATIONAL STANDARDS. *European science review* 5-6 (2019), 52-54.
- Pollak, Michael & Ebner, Martin. (2019). The Missing Link to Computational Thinking. *Future Internet*. 11. 263. 10.3390/fi11120263.
- Potgieter, I. and Coetzee, M. (2013) 'Employability attributes and personality preferences of postgraduate business management students', *SA Journal of Industrial Psychology*. doi: 10.4102/sajip.v39i1.1064.

- Powell, R.A., Single, H.M. and Lloyd, K.R., 1996. Focus groups in mental health research: enhancing the validity of user and provider questionnaires. *International Journal of Social Psychiatry*, 42(3), pp.193-206.
- Priestley, M. and Minty, S., 2013. Curriculum for Excellence: 'A brilliant idea, but...'. *Scottish Educational Review*, 45(1), pp.39-52.
- Professional Standards for Teachers - The General Teaching Council for Scotland (gcs.org.uk) Available at <https://www.gcs.org.uk/professional-standards/professional-standards-for-teachers/>
- Raquel Hijon-Neira, Diana Perez-Marin, Pizarro C, Connolly C. The Effects of a Visual Execution Environment and Makey Makey on Primary School Children Learning Introductory Programming Concepts. *IEEE Access* 2020 01/01;8:217800-217815
- Rich, P.J., Bartholomew, S., Daniel, D., Dinsmoor, K., Nielsen, M., Reynolds, C., Swanson, M., Winward, E. and Yauney, J., 2022. Trends in tools used to teach computational thinking through elementary coding. *Journal of Research on Technology in Education*, pp.1-22.
- Rose, J., 2009. Independent review of the primary curriculum (England).
- Ryan, G.W. and Bernard, H.R., 2003. Techniques to identify themes. *Field methods*, 15(1), pp.85-109.
- Salac, J. Thomas, C. Butler, K and Franklin D. 2021. Supporting Diverse Learners in K-8 Computational Thinking with TIPP&SEE. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 246–252.
- Salac, J., Thomas, C., Butler, C. and Franklin, D., 2021, June. Understanding the Link between Computer Science Instruction and Reading & Math Performance. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1* (pp. 408-414).
- Salac, J., 2020, August. Diagramming as a Strategy for Primary/Elementary-Age Program Comprehension. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (pp. 322-323).
- Salac, J. and Franklin, D., 2020, June. If they build it, will they understand it? exploring the relationship between student code and performance. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 473-479).t al., 2020)
- Salac, J. Cathy Thomas, Bryan Twarek, William Marsland, and Diana Franklin. 2020. *Comprehending Code: Understanding the Relationship between Reading and Math*

- Proficiency, and 4th-Grade CS Learning Outcomes. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education. 268-274.
- Sanzo, A., Schapachnik, F., Factorovich, P. and O'Connor, F.S., 2017, October. Pilas bloques: A scenario-based children learning platform. In 2017 Twelfth Latin American Conference on Learning Technologies (LACLO) (pp. 1-6). IEEE.
- Sapounidis, T. and Demetriadis, S., 2013. Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences. *Personal and ubiquitous computing*, 17(8), pp.1775-1786.
- Sapounidis, T., et al 2015 Demetriadis, S. & Stamelos, I. Evaluating children performance with graphical and tangible robot programming tools. *Pers Ubiquit Comput* 19, 225-237 (2015). <https://doi.org/10.1007/s00779-014-0774-3>
- Schaberg, K., 2019. Teaching Soft Skills in Workforce Programs: Findings from WorkAdvance Providers. American Enterprise Institute.
- Schulte, C 2008. Block model: an educational model of program comprehension as a tool for scholarly approach to teaching. In Proceedings of the Fourth international Workshop on Computing Education Research. 149-160.
- Schulte, C., 2013, November. Reflections on the role of programming in primary and secondary computing education. In Proceedings of the 8th Workshop in Primary and Secondary Computing Education (pp. 17-24).
- Scotland, E., House, D., Park, A.B. and Way, A., 2015. How good is our school?. Glasgow: Education Scotland. Retrieved.
- Scottish Executive (2004a) A Curriculum for Excellence The Curriculum Review Group, Edinburgh: Scottish Executive
- Scottish Government. 2015. Improving Schools in Scotland: An OECD Perspective.
- Scottish Index of Multiple Deprivation 2020 - gov.scot (www.gov.scot) available at https://www.gov.scot/collections/scottish-index-of-multiple-deprivation-2020/?utm_source=redirect&utm_medium=shorturl&utm_campaign=simd accessed 02/01/2022
- Sebba, J., Kent, P. and Tregenza, J., 2012. Joint Practice Development: what does the evidence suggest are effective approaches. Nottingham: National College of School Leadership (NCSL).
- Sentance, S. and Waite, J., 2017, November. PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. In Proceedings of the 12th Workshop on Primary and Secondary Computing Education (pp. 113-114).
- Sentance, S., Csizmadia, A. Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Educ Inf Technology* 22, 469–495 (2017)

- Sentance, S. and Waite, J., 2018. Computing in the classroom: Tales from the chalkface. *It-Information Technology*, 60(2), pp.103-112.
- Sentence S, Jane Waite, and Maria Kallia. 2019. Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education* 29, 2-3 (2019), 136-176.
- Sentance TOCE Project 2015
- Serafini G. (2011) Teaching Programming at Primary Schools: Visions, Experiences, and Long-Term Research Prospects. In: Kalaš I., Mittermeir R.T. (eds) Informatics in Schools. Contributing to 21st Century Education. ISSEP 2011. Lecture Notes in Computer Science, vol 7013. Springer, Berlin, Heidelberg.
<https://doi.org/10.1007/978-3-642-24722->
- Sermsuk, S., Triwichtkhun, D. and Wongwanich, S., 2014. Employment conditions and essential employability skills required by employers for secondary school graduate. *Procedia-Social and Behavioral Sciences*, 116, pp.1848-1854.
- Serrat, O., 2017. The five whys technique. *Knowledge solutions: Tools, methods, and approaches to drive organizational performance*, pp.307-310.
- Shah, Vipul. (2019). CSpathshala: bringing computational thinking to schools. *Communications of the ACM*. 62. 54-55. 10.1145/3343445.
- Shailaja, J. and Sridaran, R. (2015) 'Computational Thinking the Intellectual Thinking for the 21st century.', *International Journal of Advanced Networking & Applications*, May 2015 Special Issue.
- Sharing Argyll Learning ideas available at Kodu 6-Week Teaching Pack | SALi Accessed (glowscotland.org.uk) on 31/12/2021
- Scherer, R., Siddiq, F. and Viveros, B.S., 2020. A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Computers in Human Behavior*, 109, p.106349
- Shulman L. 1986. Those who understand: Knowledge growth in teaching. *Educational researcher* 15, 2 (1986), 4-14.
- Shulman, L. S. (1986a). Paradigms and research programs in the study of teaching: A contemporary perspective. In M. Wittrock (Ed.), *Handbook of Research on Teaching* (3 ed., pp. 3-36). New York: Macmillan Publishing Company.
- Shulte, C., 2008. Block Model an educational model of program comprehension as a tool for a scholarly approach to teaching.
- Shvarts A and Arthur Bakker. 2019. The early history of the scaffolding metaphor: Bernstein, Luria, Vygotsky, and before. *Mind, Culture, and Activity* 26, 1 (2019), 4-23.

- SICSA Computing Science at School. 2017. TeachCS Scot, Teach Computer Science.
<http://teachcs.scot/>
- Silver J L (1982) Predicting Success in a First Programming Course. SIGSCE '82 Proceedings of the thirteenth SIGCSE technical symposium on computer science education, 147-150.
- Sima, V. et al. (2020) 'Influences of the industry 4.0 revolution on the human capital development and consumer behavior: A systematic review', Sustainability (Switzerland), 12(10). doi: 10.3390/SU12104035.
- Sirin SR (2005). Socioeconomic status and academic achievement: A meta-analytic review of research. *Review of Educational Research*, 75, 417–453.
 10.3102/00346543075003417 [CrossRef] [Google Scholar]
- Smith, N., Sutcliffe, C. and Sandvik, L., 2014, March. Code club: bringing programming to UK primary schools through scratch. In Proceedings of the 45th ACM technical symposium on Computer science education (pp. 517-522).
- Stolee, K.T. and Fristoe, T., 2011, March. Expressing computer science concepts through Kodu game lab. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 99-104).
- Stone, Harold S. (1972). Introduction to Computer Organization and Data Structures (1972 ed.) (p.4). McGraw-Hill, New York. ISBN 978-0-07-061726-1.
- Strickland, D.S. and Riley-Ayers, S., 2006. Early literacy: Policy and practice in the preschool years. *Preschool policy brief*, 10(4), pp.1-12.
- Szabo, C., Sheard, J., Luxton-Reilly, A., Becker, B.A. and Ott, L., 2019, November. Fifteen years of introductory programming in schools: a global overview of K-12 initiatives. In Proceedings of the 19th Koli Calling International Conference on Computing Education Research (pp. 1-9).
- Tang, K. Y., Chou, T. L. and Tsai, C. C. (2020) 'A Content Analysis of Computational Thinking Research: An International Publication Trends and Research Typology', Asia-Pacific Education Researcher. doi: 10.1007/s40299-019-00442-8.
- Tedre, M., 2022, October. Computational Thinking 2.0. In *Proceedings of the 17th Workshop in Primary and Secondary Computing Education* (pp. 1-2).
- Tsarava, K., Moeller, K., Román-González, M., Golle, J., Leifheit, L., Butz, M.V. and Ninaus, M., 2022. A cognitive definition of computational thinking in primary education. *Computers & Education*, 179, p.104425.
- The University of Edinburgh. (n.d.). Developing and implementing a new computing science curriculum in Scottish schools. [online] Available at:

- <https://www.ed.ac.uk/education/rke/making-a-difference/new-computing-science-curriculum> [Accessed 24 Dec. 2021].
- Training, A. G. D. of E. and (2019) A Student Focused National Career Education Strategy: Ready for a World Yet to Be Imagined, Australian Government Department of Education and Training.
- Tsarava, K., Moeller, K., Román-González, M., Golle, J., Leifheit, L., Butz, M.V. and Ninaus, M., 2022. A cognitive definition of computational thinking in primary education. *Computers & Education*
- Twinkle resource accessed on 23/02/23 at <https://www.twinkl.co.uk/resource/t2-i-85-catch-the-bugs-scratch-activity-sheet>
- Van Tassel- Baska, Quek, & Feng, 2007, p. 85). Bracken, B.A., VanTassel-Baska, J., Brown, E.F. and Feng, A., 2007. Project Athena: A tale of two studies. Overlooked gems: A national perspective on low-income promising learners, pp.63-67.
- Velázquez-Iturbide, J.Á., Paredes-Barragán, P. and Urquiza-Fuentes, J., 2022, October. An Experience in Explicitly Training Pre-Service Early Childhood Teachers in Programming Concepts with ScratchJr. In *Proceedings of the 17th Workshop in Primary and Secondary Computing Education* (pp. 1-2).(Paper 7)
- Vickery A. 2013. Developing active learning in the primary classroom. Sage.
- Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools, *Computers & Education*, Volume 97, 2016, Pages 129-141, ISSN 0360-1315,
- Voskoglou, M. G. and Buckley, S. (2012) 'Problem Solving and Computational Thinking in a Learning Environment', (May 2014). Available at: <http://arxiv.org/abs/1212.0750>.
- Waite, J., 2023. The Role of Design in Primary (K–5) Programming. *Computer Science Education: Perspectives on Teaching and Learning in School*, p.237.
- Waite, J., 2017. Pedagogy in teaching computer science in schools: A literature review. *London: Royal Society*, 253.
- Waite, J., Curzon, P., Marsh, W., and Sentence, S. 2016. Abstraction and common classroom activities. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education (WiPSCE '16)*. Association for Computing Machinery, New York, NY, USA, 112–113.
- Webster, A., McNeish, D., Scott, S., Maynard, L. and Haywood, S., 2012. What influences teachers to change their practice. A rapid research review. Centre for Understanding Behaviour Change.
- Wing J. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33-35.

- Wing, J. M. (2008) 'Computational thinking and thinking about computing', *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*.
- Wilson, B & Shrock(2001) Contribution to success in an introductory computer science course: a study of twelve factors, SIGCSE '01 Proceedings of the thirty-second SIGSCE technical symposium on Computer Science Education.
- Wolfgang, C., Stannard. L., et al 2001. Block Play Performance Among Preschoolers As a Predictor of Later School Achievement in Mathematics, *Journal of Research in Childhood Education* 15:2 173-180
- Wong-Aitken, D., Cukierman, D. and Chilana, P.K., 2022, July. " It Depends on Whether or Not I'm Lucky" How Students in an Introductory Programming Course Discover, Select, and Assess the Utility of Web-Based Resources. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education* Vol. 1 (pp. 512-518).
- World Economic Forum, 2016. The future of jobs: Employment, skills and workforce strategy for the fourth industrial revolution. *Global Challenge Insight Report*.
- Yeni, S., Grgurina, N., Hermans, F., Tolboom, J. and Barendsen, E., 2021, October. Exploring teachers' PCK for computational thinking in context. In *The 16th Workshop in Primary and Secondary Computing Education* (pp. 1-10). Paper 5

Appendices

Appendix	Title	
A	Focus Group Pre-reading material	
B	Focus group questions	
C	Quadrant B - Online survey	
D	School A teacher assessment criteria	
E	School B,C,D teacher assessment criteria	
F	Curriculum for Excellence CS 3 step explained	
G	Quadrant C - Ambassador's Abbreviated lesson plans	
H	Ambassador's full lesson plans	
I	Sample learning journal P4	
J	Quadrant D - Children's Toys and Games survey	
K	P4 Sample table of results for P4 School B	
L	Children's view of level of difficulty -reflections	
M	CfE Computing Science 3 Step overview	

Appendix A Focus Group Pre-reading material

Background: The digital revolution is about our ever-increasing ability to identify and automate information processes. Some information processes can be carried out by humans as well as computers, remember that the original computers were humans carrying out complex mathematical tasks or procedures. This blend of computing and human activities suggests that the twin concepts of information and process appear central to computing but also to a broad range of other work contexts.

Information: This skill relates to the way individuals make sense of, or interpret, the information to which they are exposed. The digital revolution brings a different type of problem solving with process automation or information processing using automation. In recent years the term information processing has often been applied to computer-based operations specifically. In computer science, it is a process that transforms information into another form. Information appears in the guise of data and its representation and organisation. Conceptualising problems in understanding before solutions can be made requires information processing. Creating data representations to hold information for example, to create algorithms requires the skill of extracting relevant information.

Process: An algorithm is a description of a process with special properties and they vary in complexity. As a concept, in its own right, process, does not typically appear but it is clearly related to the concept of an algorithm which lies very much at the heart of CS curricula. For example, the described process should have a well-defined starting point and proceed through discrete steps, reaching a clear completion point in a finite time⁹, accomplishing a well-defined task; the description should be unambiguous, and have a level of generality in it such that it can be used to solve a whole class of tasks. Furthermore, the first algorithms, coming to us from historical times, defined information processes, involving numbers and arithmetic, such as Euclid's Algorithm or the Sieve of Eratosthenes. Processes have more general characteristics¹⁰. For example, they may consist of discrete observable steps or else they may be continuous; they may operate on information or on physical artefacts, such as in an industrial process; they may not complete or be well-defined, and so on. There are many shared characteristics too, such as

⁹ NB: An algorithm need not be finite and may also be ambiguous.

¹⁰ NB: This example of generality is more than pattern making.

parallel and communicating processes perhaps with shared resources; they may be hidden or visible activities; there needs to be some form of executing agent that drives the process forward, whether that is human or mechanical. In order to appreciate a good algorithm, it is also important to understand what is not a good algorithm, and that requires a broader understanding of the realm of processes generally.

Modelling: Modelling in the simplest terms is the ability to represent some part of the real world through a computer. Modelling is defined as creating visual representations of real objects as an information process. It describes a process or information in another format through decomposition, generalisation, abstraction, and pattern-recognition. Once the correct level of detail is extracted, a model is created. In the CS domain, the abstraction requires an understanding of technology to create the representation. Typically programming attempts to model reality and create a system which can be used as a surrogate for another system. It is a fundamental idea and concept of computer science. To avoid this becoming overly general, it is important that the CS problem solving domain is clear, and this is the domain of problems that can be represented as information processes. Solving a problem computationally requires the skills of analysing the problem in terms of the information and processes involved and then modelling these using the computational tools available, whether this be a programming language, or database system, or any other computational platform. As in other problem-solving contexts, the key is the ability to build a model of the problem using the modelling tools at hand. Considering typical CT terms, decomposition, generalisation and pattern-matching are all part of this problem-solving or modelling activity that is focused on information and processes.

Reasoning: In computing science, a crucial problem-solving skill is the ability to reflect on or reason about the model or solution that has been created. Mental, written or computational models need to be tried and tested to ensure the best fit for information processes that they represent. Often there is more than one solution to a given problem, therefore the ability to predict the outcomes of the algorithm designed to solve a problem means that the best solution can be found. Given the precise nature of executing programs on a machine, there is a need for accuracy. Reasoning through analysis and refinement is essential to detect and correct errors in algorithms and programs. Reasoning requires valid arguments that can be defended rigorously or executed on a machine (Huth and Ryan 2004). This emerges in testing and refining computational artefacts and is also an essential part of debugging.

Appendix B Focus Group questions

- 1) What is your background in education?
- 2) Have you directly taught introductory programming?
- 3) Does everyone achieve success in their first exposure to introductory programming?
- 4) What factors contribute to children's success in introductory programming in school?

Appendix C Quadrant B - Online survey

Pre-requisites for computational thinking: Survey content

Summary

The survey aims to explore the question, “Does every pupil show proficiency in programming when it is first introduced in class?” For validity, the survey will be issued to around 100 practitioners. There will be an opportunity for any additional comments or burning issues you feel worth consideration in relation to any aspect of computational thinking. Findings will underpin the creation of a short series of activities for primary age pupils.

Background

Computer Science (CS) education has been hailed as valuable to everyone in the population, and if so, it should be taught to all. In addition, The Commission for Developing Scotland’s Young workforce (DYW) (2014) made recommendations to improve the employment prospects for Scotland’s young people. With the increasing value of computational thinking from governments and industries, including non-digital, there is a need for high quality learning and teaching in the area.

A typical reaction to this need is to equate computing science skills with programming skills and introduce programming from very early in the school curriculum. Given the difficulty that both schools and universities have faced for decades in enabling *all* learners to succeed at programming, the value of this approach is being questioned. In particular, are there pre-requisite skills to programming / computational thinking?

As a starting point, there is a need to ascertain whether novices who are introduced to programming via one of the new wave of educational programming environments (e.g. Scratch, Alice, Kodu, Bebots etc.) are broadly universally successful in picking up the ideas and skills.

If all novices are successful, then a curriculum with exposure to programming at the early stages is sound. But if they are not, then there is a need to explore why some novices succeed where others don't. The difference could be innate, but our hypothesis is that it is more likely to be to do with attitudes to learning, kinds of toys played with, parental and sibling role models. The data is gathered at the end of the questionnaire.

This research is part of a PhD study exploring worthwhile computational thinking skills irrespective of a child or young person’s chosen career.

Questionnaire (1)

Does every pupil show proficiency in programming when first introduced in school?

Context

- 1) Have you directly taught programming to pupils? yes/no
2) Was it their first formal education of programming? yes/no/don't know
3) Which programming environment did you use? Kodu
Scratch
Alice
Other _____ Please specify

4) What age were the pupils? _____

5) Which teaching approach was used? Tick all that apply

- A) Demo in front of class
B) Follow the leader step by step alongside the teacher
C) Worksheet of instructions
D) Directing a small group with others working independently.
E) Combination of above (indicate which).

- 6) Ratio of pupils to devices A) 2 children with one device
B) 1 child with one device
C) other _____

7) Which concepts were taught?

Levels of success

- 8) How many pupils were taught at the same time? _____
I. How many pupils were highly motivated before and during the task? _____
II. How many pupils showed increased motivation as the task continued? _____
III. Did any pupils show unexpected levels of motivation in relation to their motivation levels in other aspects of their learning? _____

9) **How many pupils (approx.) achieved immediate success?** _____

(Pupils who required no or minimal instruction and completed the task independently)

10) **How many pupils (approx.) achieved some success?** _____

(Pupils who required some support, worked steadily through the activity and maybe sought help from peers or adults)

11) **How many pupils (approx.) needed high levels of support?** _____

(Pupils who did not complete the activity, were distracted or spent over long periods of time on one aspect of the task)

12) How many pupils continued to work on developing their skills through the programming environment in their own time? _____

External factors

9) After the initial introduction how many pupils continued to develop their skills at home or in their own time? _____

10) Were pupils aware that they could develop their skills at home or out of school? (EG; downloading the programming environments to their own devices etc) _____

11) Please write other interests and or characteristics that you are aware of that the more successful pupils have? (Please leave blank if not sure)

12) Please note any interests or characteristics of pupils needing high levels of support that you are aware of (Please leave blank if not sure)

Thank you very much for completing this survey. If you are interested in continuing to work with us for further research into this important area please tick this box and add your contact details below.

Email address _____ (Please write clearly and repeat) _____

Twitter _____

Optional

Additional section for participants interested in continuing to be involved.

The purpose of this table is to profile the characteristics of the most successful pupils and those needing the most support in learning programming. This will enable further research to enable all pupils to be successful computational thinkers. Please add any comments that you feel you can answer next to each category. There is no need to complete all of the table.

Is there standardised assessment data for pupils? Yes / No/Don't know

Achievements in literacy of the most successful pupils in programming.		Achievements in literacy of pupils needing the highest levels of support in programming.	
Reading		Reading	
Writing		Writing	
Listening		Listening	
Talking		Talking	
Achievements in numeracy and mathematics of the most successful pupils in programming.		Achievements in numeracy and mathematics of pupils needing the highest levels of support in programming.	
Mental agility		Mental agility	
Problem solving		Problem solving	
Number, money measure		Number, money measure	
Shape, position movement		Shape, position and movement	
Information handling		Information handling	
Interests and play preferences of the most successful pupils in programming.		Interests and play preferences of pupils needing the highest levels of support in programming.	
Play preferences		Play preferences	
Interests		Interests	
Motivation levels to learning across curricular areas.		Motivation levels to learning across curricular areas.	

Appendix D School A

School A: 6 week introductory course Lego Wedo Success criteria.

Success criteria (SC) for each lesson. Educators award 1 point for each assessed element of the course.

Week 1

1. Demonstrated skills in using software
2. Took part in discussion and debate
3. I can use icons to programme a 3D model to make it do different things.
4. I can explore software and related materials to discover what they can do.
5. I can assess my groups learning.

Week 2

1. Demonstrated skills in using software
2. Took part in discussion and debate
3. I can use icons to programme a 3D model to make it do different things.
4. I can explore software and related materials to discover what they can do.
5. I can assess my groups learning.

Week 3

1. Demonstrated skills in using software
2. Took part in discussion and debate
3. I can use icons to programme a 3D model to make it do different things.
4. I can explore software and related materials to discover what they can do.
5. I can construct a drumming monkey using either the instructions given or design my own.

Week 4

1. Demonstrated skills in using software
2. Took part in discussion and debate
3. I can use icons to programme a 3D model to make it do different things.
4. I can use my learning to construct 3D objects with moving, programmable parts.
5. I understand how machines and computers process information with hands on learning.

Week 5

1. Demonstrated skills in using software
2. Took part in discussion and debate
3. Evaluating products I can adapt and improve my programmable model building through trial and error and discussion.
4. I can use icons to programme a 3D model to make it do different things.
5. I can use my learning to construct 3D objects with moving, programmable parts.

Week 6

1. Demonstrated skills in using software
2. Took part in discussion and debate
3. Evaluating products I can adapt and improve my programmable model building through
4. I can use icons to programme a 3D model to make it do different things.
5. I can use my learning to construct 3D objects with moving, programmable parts.

Appendix E - Quadrant B Assessment criteria

School B,C,D Weekly assessment criteria(SC).

Teachers award 1 point for each assessed element of the course. 6 week assessment by teacher using this criteria for each lesson.

Kodu – Weekly planning sheet

Week 1

1. Become familiar with the design tools
2. Understand how to move the terrain
3. Understand how to paint the terrain
4. Understand how to design a visually appealing environment and add objects to the terrain (such as trees, apples etc.)
5. Be able to program a character to move around the terrain.

Week 2

1. Explore addition terrain design features
2. Program the character to interact with objects and characteristics such as:
 - Eat apples function
 - Program Bump objects function
 - Characters jump function, Run faster

Week 3

1. Programming one characters to move and interact with the environment.
2. Programming multiple characters to move and interact with the environment.
3. Program the characters to talk (via pop up text boxes)
4. Start to build a storyline for your game (this may be on paper)
5. Understand how to use paths

Week 4

1. Create a flying saucer game (which involves a flying saucer moving on a path and dropping apples)
2. Understand random function in Kodu to drop multicolored apples
3. Understand how to use Timers.
4. Understand how to use Counters.

5. Use two of three new features.

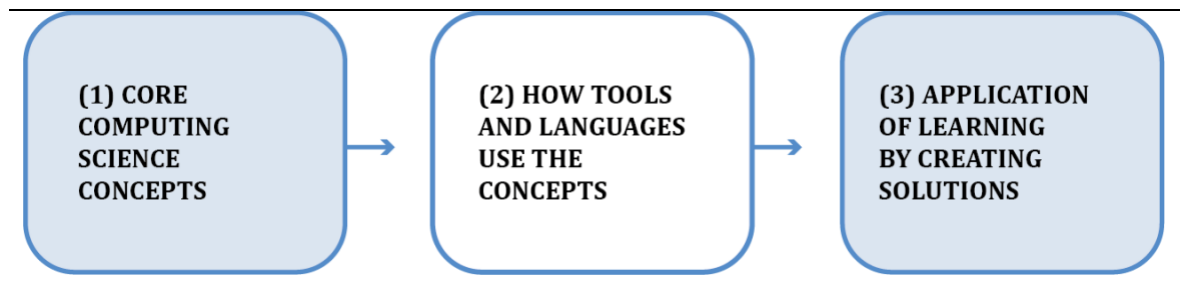
Week 5

1. Spawning (automatically generating additional characters)
2. Building a race game.
3. New Level
4. Action functions
5. Add two of three new features to the game.

Week 6

1. Win/Lose Condition (making the game end when certain conditions are met)
2. Using the teleport function
3. Pupil showcase game and ideas to class
4. Peer Assessment accurately
5. Game can be played by peers and function as expected.

Appendix F Curriculum for Excellence 3-step explained.



The Scottish CS Curricula Framework organised into a 3-step approach.

The CS hierarchical organisers start with:

- (1) conceptual knowledge gained from ‘understanding the world through CS’ to then
- (2) understanding computational languages and technologies before children can
- (3) design build and test computing solutions.

This 3-step approach outlined in Fig 2 is based on a program comprehension educational model that aligns with the Scottish Curriculum for Excellence CS experiences and outcomes. Its philosophy is that children should not be expected to write correct programs/scripts/instruction (step/organiser 3) without the knowledge and understanding of the underlying concepts (step/organiser 1) or (step/organiser 2) the ability able to accurately read and understand programs in that language.

The 3-step framework specifies that CS specific skills and knowledge are necessary before learners can successfully problem solve and build solutions. It does not mean that children must gain an understanding of all concepts in organiser one and all languages and tools in organiser two before developing and building solutions in organiser three. The organisers can be covered in one lesson sequentially. The approach is applicable at all levels of difficulty albeit introductory programming course or quantum computational modelling. “It is a spiral curriculum where learners will revisit concepts at increasing depth.” (Farrell, et al 2017). The CS curriculum requires more than repetition. It requires deepening of knowledge with each successful encounter building on the other.

Appendix G Quadrant C Ambassador's lesson plans

LESSON ONE

Part one

Children describe processes in everyday life such as going to school. Children watch a 'fun' hand jive video and describe the process they are observing, they then follow the hand jive.

Part two

The focus on how computers process information involves children working in groups of four. learn about and act out parallel and sequential processes using the green flag and red stop sign from Scratch Jnr. Children identify the sequential and parallel processes in an upbeat video clip and in a visual representation of making breakfast.

Part three

Continued focus on process descriptions. Children play program the Ambassadors by reading pre-written instructions to them. The Ambassadors follows the instructions and children describe what is happening. In groups they create new descriptions for Ambassadors to follow and improve on the original instructions. Predicting what will happen.

Part four

Improving process descriptions. Children revisit the hand jive and create a written description of the process for each other to follow.

LESSON TWO

LI2.1 LI2.2 SC2.1 SC2.2 SC 2.3 SC 2.4

Part one

Children revisit lesson one using Scratch Jr tiles displayed on the whiteboard to reinforces the physical representation of the Scratch Jnr commands. They play a competitive game for points, observing a few Scratch animations and predict what is going to happen.

Part two

Children revisit program the teacher in the context of how computers process information. Using oversized Scratch Jnr tiles they predict, create and optimise precise instructions to follow.

Part three

As a teaser, children watch a few of the animations that they will be working with in lesson 3. They describe what's happening in each animation.

LESSON THREE

LI: 3.1, 3.2,,3.3 SC: SC3.1, SC3.2, SC3.3

Part one

Children revisit the Scratch Jnr tiles explaining what each tile and groups of tiles instructs the computer to action. Using the iPad as a visualiser revisit some of the children's instructions for the maze game that they created for the teacher in lesson two.

Part two

Children watch a few animations and describe and predict what's happening using the paper icons. The Ambassadors show the scripts for each of the animations and shows the children how to modify the project. At all times children explain their scripts before actioning.

Part three

Revisit the term optimisation and explain how to create a game. Children take selfies to personalise the sprites. They modify the scripts that they are familiar.

Appendix H Ambassador's full unedited lesson plans

These are created by the final year CS undergraduate students (Ambassadors)

Concepts of SAL3 that need to be covered:

- Processes
- Sets of instructions
- Understanding processes in sequences
- Understanding processes in parallel
- Start and end of the process
- Repetition
- Evaluation
- Optimisation

Set of instructions:

SAL 1: *Breakfast* - breaking large problem into smaller ones

Understanding processes in sequences:

SAL 1: *Instruction stations* - children are given instructions

SAL 2: *Simon Says* - children are executing a set of instructions when Simon says to do so

Understanding processes in parallel:

SAL 1: *Instruction stations* - split the children into groups of 4

Start and end of the processes:

SAL 1: *Instruction stations* - Show the children green and red flags for starting and stopping

SAL 2: *Simon Says* - if 'Simon Says,' to do something, then everyone does that thing, but if Simon doesn't say, then participants do nothing

Repetition:

SAL 1: *Program the Teacher / Classmate*

Evaluation:

SAL 3: *Game*

Optimisation:

SAL1 : *Program the Teacher/ Classmate*

SAL3: *Game*

Learning intention: To understand what processes are and process descriptions

Success Criteria:

SC1 - I can carry out the steps of a process description as they are given to me

SC2 - I understand that a process description specifies the steps of a process

SC3 - I can describe orally each step of a process that I do in every day life

SC4 - I can create and improve a written or graphical process description

Assessment – (embedded within the lesson)

SC1) Children observe each other undertaking a process and traffic light each other's responses
(self-assessment) (*hand jive*)

SC4) Children create set of instructions for the teacher to follow and evaluate and improve them.
(*Program the Teacher*)

Scale for assessment

5 Complete achievement of the goal, task or understanding

4 Mostly complete achievement of the goal task or understanding

3 Partially complete achievement goals

2 Very incomplete

1 Did not complete any part of the goal, task or understanding

0 did not attempt/ Other

SAL1

The aim of the lesson is to understand what processes are, and process descriptions

Introduction (): 5 mins

- The lesson should begin with the teacher introducing him/herself to the class. The teacher should explain why s/he would like to teach the students about programming. S/he should briefly ask students what they know about programming.

What do you know about computers?

Part one: Processes in every day life 10 - 15 minutes

SC 1: I can carry out the steps of a process as they are given to me

‘Computational Thinking’

- Computational thinking is not thinking about computers or like computers. Computers don’t think for themselves. Not yet, at least! Computer scientists are interested in finding the most efficient way to solve problems. They want to find the best solution that solves a problem correctly.
- What is a process? * *series of actions and step to achieve a goal* *
- For example, putting my shoes on is a process. Do you think walking is a process.
- What about the process you are doing?
- Can anyone tell me their process from waking up to arriving at school?
- We are going to try out a process. Let’s watch this clip of a hand jive
- <https://www.youtube.com/watch?v=eIozjNOgrm8>

SC1 - In pairs let’s try the hand jive. Did everyone manage it?

CHILDREN COMPLETE SELF ASSESSMENT LEARNING JOURNAL

Let’s think of some other processes that you do?

(Washing hands? Getting dressed in the morning? Lining up for school lunches?)

Part two: Computer processes

Computers and people are very similar, because both execute processes! You are now going to represent a computer process.

Instruction Stations (): 5 mins

Split the class into four groups and assign them to four different stations. Each station will correspond to an instruction to follow (e.g. clap your hands, stomp your feet, jump up and down, tap your hands on your head). When the teacher raises the green flag card, students follow the instruction at their station. They stop when the teacher raises the red stop sign card. Students should then rotate to a different station. Repeat this activity until all students have moved through each station once. The teacher should explain how the green flag signifies the start of a program, while the red stop sign signifies the end of a program.

Materials: Green flag card, red stop sign card

I'm going to introduce words that a computer scientist would use. Parallel and sequential processes. Can anyone guess what these are? Turn to your shoulder partner and guess what they could be? (Show words on the board)

A process can be parallel and sequential.

Let's repeat that - parallel and sequential.

But what does parallel process mean?

It is when we are doing two processes in the same time - e.g. I am walking and talking on my phone. What parallel processes do you do? (e.g. playing on the PC while listening to music on the PC)

And, what is a sequential process?

It is when we do one process after the other!

So, before I came into this class, I walked to the door and THEN I opened it.

Turn again to your shoulder partner and together think of any sequential processes that you do?

Ask for their responses

- Getting out of bed and putting your socks on, maybe?

Show part of the Rube Goldberg Machin youtube video to see parallel and sequential processes.

<https://www.youtube.com/watch?v=qybUFnY7Y8w>

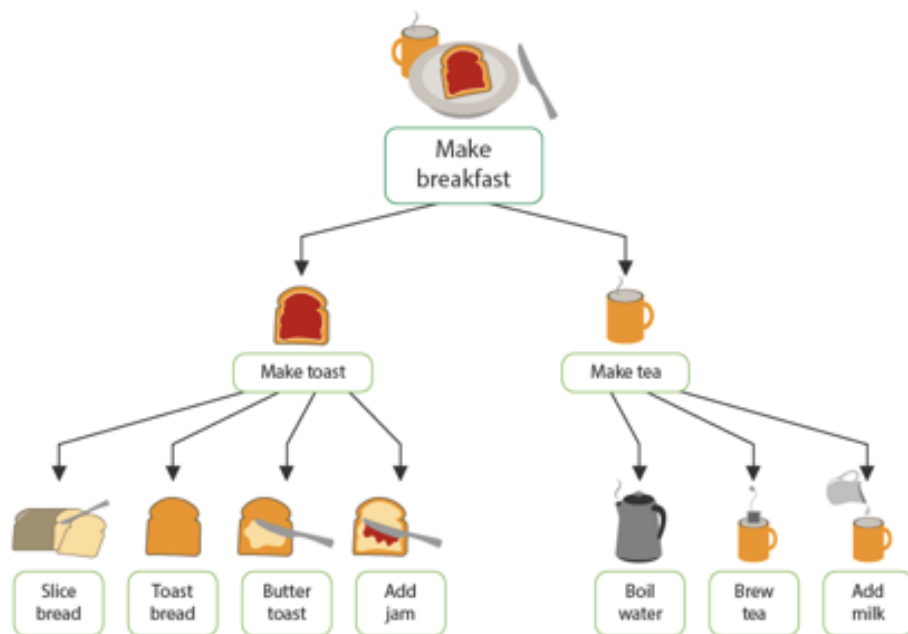
Part three:

Part three: Breaking down processes **5 mins**

- Computing Scientists break down a large process into smaller baby processes.

What about the process of making a breakfast? In groups discuss what is happening in each picture.

– Children report back the process taking place in their group Can you say what would happen for this to be a parallel or sequential process (Challenging but they may manage it)



Part four: Improving processes

Program the Teacher (): 10 mins

Split the class into groups (tables) and give each one a different object to write instructions to from your starting point. Then try out every set of instructions to see if they are correct and they are leading you to the object. If not ask the students to improve their instructions.

(SC4) One student reads (read) a set of instructions that the teacher has to follow. Children turn to their shoulder partner and predict where in the classroom the teacher will end up when they follow these instructions. The teacher follows the instructions.

The children now become responsible for directing their teacher to a specific location in the classroom. However, during this lesson, students will only be able to use a specific set of possible instructions instead of simply using plain English. Examples of these specific instructions are:

- Step forward
- Step backward
- Turn right
- Turn left
- Turn until you see something

They should instead say, “Move forward ____ steps.” When sequences of instructions do not work (perhaps because the number of steps taken were incorrect), students should alter their instructions. After the activity is over, the teacher should discuss how important it is to be precise and how important order is in programming.

SC4 COMPLETE SELF ASSESSMENT STUDENT JOURNALS

Return back to the hand jive. Ask the children how we could make it better – we could break it down into small steps. What are these?

Show the steps on the board and ask the children to have another go. Ask them if that was easier?

Plenary

Lesson name: Start from Scratch!

Learning intention:

- To apply the computational concepts introduced in the first lesson to certain activities such as ‘Simon Says’
- To introduce pupils to the basics of ScratchJr visual programming language and predict the outcome of basic predefined examples.

Requirements for the lesson:

- Teacher's iPad should have ScratchJr installed and be ready for usage.
- Scenarios for Script Prediction should be loaded on the iPad
- A live scenario or a video recording of one of the scenarios from SAL3
- Printouts:
 - **18 sheets of Commands ,**
 - **1 sheet of GreenCrosses,**
 - **1 sheet of RedCross+Compass,**
- 6 mini whiteboards or 6 white sheets of paper
- Ask the teachers to set up a class without any tables if possible.

Part one: Recap**Warm-up Activity 5 mins**

Play a modified version of 'Simon Says' with the group. If 'Simon Says,' to do something, then everyone does that thing, but if Simon doesn't say, then participants do nothing. You can use the ScratchJr blocks on the PowerPoint slides to show an action.

Characters in ScratchJr behave in the same way; they need to be triggered before they will begin their animations. This activity aligns nicely with activities in SAL1 and allows children to get the physical representation of the commands of ScratchJr.

Part two: Introduction to ScratchJr

Presentation of ScratchJr and Explanation of the Blocks **5 mins**

“So the blocks you’ve just seen on the board were from the programming language, called ScratchJr.”

“With ScratchJr, you can program your own interactive stories and games. In the process, you learn to solve problems, design projects, and express yourselves creatively on the computer.”

Run through the commands on the PowerPoint one by one and explain what they do, showing the printed out ones if relevant.

- Motion Blocks:
 - Blue Motion blocks cause characters to move on the screen. You can make characters move all over the screen (right, left, up, down, turn, and jump) by using the blue Motion blocks.
- Sound Block:
 - Makes a character make a sound (‘pop’ in this case).
- Change the Speed Block:
 - Makes a character change the speed with which it is executing the commands.
- Start on Green Flag:
 - The yellow Trigger blocks are what trigger/cause a program to begin. Each of the yellow Trigger blocks represents a different way in which a program can begin. The Start on Green Flag block creates a program that will begin whenever the Green Flag is tapped.
- End:
 - Makes the program stop.

ScratchJr Scenario Prediction **10-15 mins**

Split the class up into 6 groups and try to seat them at approximately same distance to the board.

Hand out the evaluation sheets.

SC1: I know the meaning of some ScratchJr blocks.

Introduce them to the fact that this is going to be a competition and make them aware that they can earn points for their team. Write the teams on the whiteboard so that you can keep track of the point count.

Using ScratchJr, show the starting scene of the playground on the board. The pupils, in groups, have 30 seconds to discuss what is going to happen and where the character will end up in the scenario.

After 30 seconds, one of the representatives from each team come up to the board and simultaneously point their finger at the spot where they think the character will end up at. Then the iPad will run the script, and they can see who got it right. **Two** points are awarded to each team for each right answer.

SC2: I can read and understand a basic ScratchJr script.

SC3: I can predict the outcomes of a basic ScratchJr script.

Program the RoboTeacher 15-20 mins

Introduce the pupils to the idea of the activity - “last week we did some simple programming of the teacher, but today we’re bringing this process much closer to the way computers do it”

takes out the blocks

“Now, just like we’ve just seen it happening in ScratchJr, we will be using these blocks to create a real scenario in this classroom!”

Setup:

- set up the classroom as the playground,
- make sure that the pupils get the understanding which way is up, down, left and right in the classroom - you can use the compass for that,
- set the green X signs as starting position for each group - **the starting point is the middle of the group,**
- set the red X sign on an object that the pupils are aiming to reach - it should be in front of the class with approximately same distance to each group, as displayed on slide 22,
- Split the blocks up so that each team gets the right amount:

- **4x move up**
 - **4x move down**
 - **4x move left**
 - **4x move right**
- 
- **1x start on green flag**
 - **1x end**
- 
- **1x turn clockwise**
 - **1x turn anti-clockwise**
- 
- **1x Sound block**
 - **1x Speed**
- 

Objective - Pupils have to work in groups and work out a way to get from the spot they are sitting at to a certain object in the classroom using the printed ScratchJr commands. The challenge can be made more difficult by setting obstacles:

- Physical obstacles that they have to make their way around - an example is shown in slide 25.
 - A guard that needs to hear the passphrase ‘pop’ in order to let you through
- You can also give them a time limit, so that they need to use the ‘Change Speed’ block.

Important: Let pupils know that they can write the amount of times that the move should be executed in the white slot below.

- ScratchJr can help make it simpler for characters to do the same thing more than once.
- If you want your character to move five steps to the right, you could connect five Move Right blocks.
- Or you could use one Move Right block and change the value of that block with the number pad. The number pad lets you decide how many times a block should repeat (up to 99).

Give them 5 minutes to come up with a solution.

Visit each group in the following manner:

- Take a picture of the solution,
- Go to the starting point,
- Execute the instructions,
- Return to the group - if they succeeded, award them **4 points** , if not, give them another chance to do it while you're visiting the other teams. If they get it right on the second round - give them **2 points**.

Pick the best team and announce them as the winners of the game!

SC4: I can create my own sequence of instructions using basic ScratchJr blocks.

Teaser 5 mins

As the closure/teaser, show the kids one of the cool SAL3 animations and tell the kids that next week they will already be able to do one of those!

SAL 3

Lesson name: ScratchJr animations and games

Learning intention:

- To apply the computational concepts as well as the ScratchJr knowledge introduced in the first and the second lesson into creating animations and games in ScratchJr
- To predict the outcome of ScratchJr animations and to introduce the pupils to optimisation and how to apply it in ScratchJr
- To understand unfamiliar scripts and to build on top of them

Requirements for the lesson:

- The school's iPads should have ScratchJr installed and ready for usage
- ScratchJr should have the two projects used in this workshop imported
- Visualiser so that the teacher can show how the scripts should look like

Part one: Recap

Warm-up. (10 minutes)

Briefly go over the ScratchJr tiles that the pupils learnt in SAL 2.

"Hi class, today we are going to create and play SOME ANIMATIONS/GAMES!"

Firstly, I will go through what we learnt last week."

Show on the board all of the tiles in ScratchJr. Ask some questions.

"What is this tile doing."

Teaser. (10 minutes)

Play the maze game which was the teaser from last week on the teacher's iPad using the visualiser and briefly go over the tiles used in the game.

Part two: Optimisation

Exercise 1. (15 minutes)

Optimise a game about reaching a cake. Pair up the pupils and explain how to load and start the project.

“Okay, let's pair up. Every two of you will work on one tablet. So, let's find the ScratchJr icon and click on it. If we click on the tiny house and then on the "Catch the cake" square you will see the game. As we already know, clicking on the green tiny flag will start it! Let's see what happens when we start it?”

Before optimising it, ask the pupils to take a photo of themselves and set it as their avatar.

“Let's make it more fun. Let's take a selfie and put it in the character!”

Show them on the board how to do it.

Instructions:

- 1) Click on the brush on the top left character box, then click on the camera.
- 2) After that click on the character's face and take the picture.

Ask the pupils to optimise the animation so that the character eats the cake before the cake disappears.

“Can you try and change the squares so that your character can eat the cake before it disappears? This is what computer scientists call OPTIMISATION! It is making something work better and faster.”

Go around the class and help the kids. There will be around 10 groups of 2. If no questions arise, go around and ask. See how they approach the task.

Exercise 2. (10 minutes)

Extend a game about a basketball cat. Show the pupils how it should look like on the screen. The pupils should extend the project so that the cat throws it to the basketball hoop and scores a point.

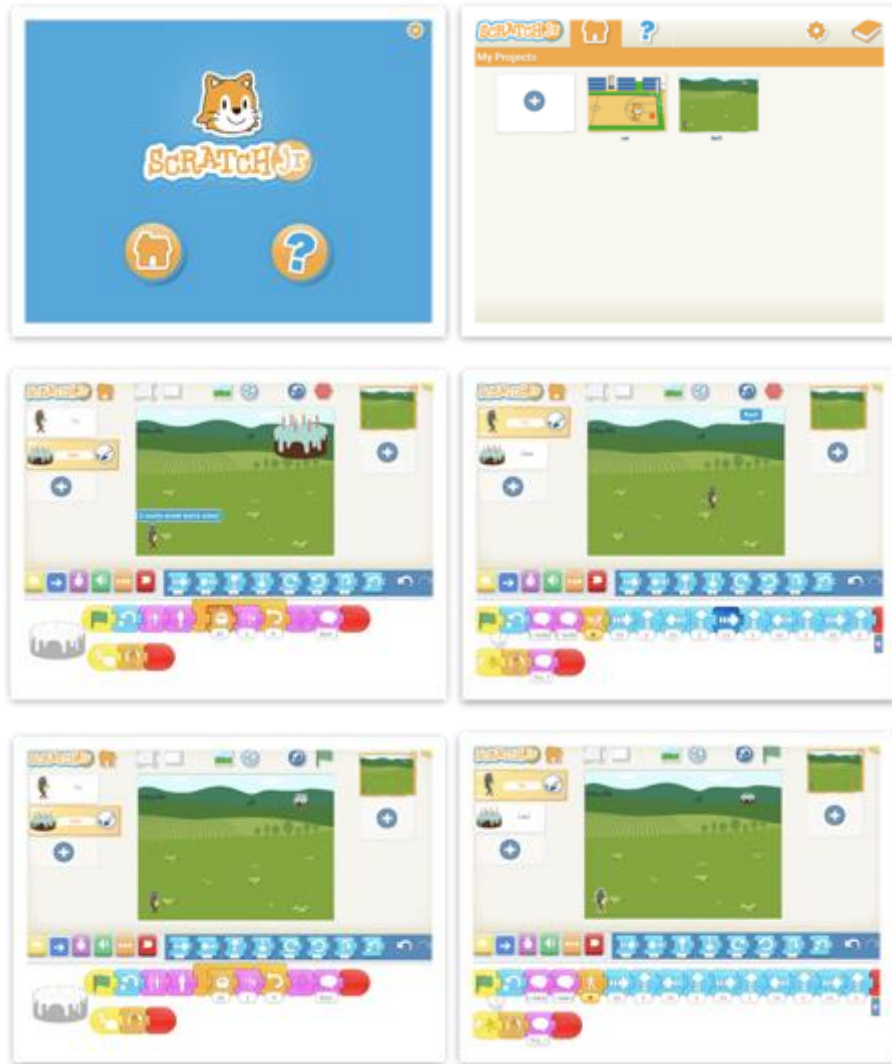
“Okay, let's see another game! If we click on the tiny house and then on the "Basketball cat" you will see it! I will show you how my animation looks like! Okay, now press the green flag again to start the script. Does not look like mine, does it! Can you make it look like my script.”

This exercise teaches the pupils to build on top of a project and to understand the way unfamiliar script works. If some students finish early ask them to add more things to the project or even ask them to create a completely new one from SCRATCH.

Walk around and check pupils' work. Ask them questions to see if they understand what they are doing. There will be approximately 10 groups of 2. Do not spend TOO much time with one group.

Print screens

Exercise 1.



Exercise 2.



Appendix I Sample learning journal

Computing Science



Name _____

Class _____

These are my two key areas for development during lesson one.

Lesson 1

1. I can carry out the steps of a  process that are given to me.

2. I can create and improve a

process description.

Lesson Evaluation

What is a process description?



A process is a sequence of steps with a beginning and end (state) that completes a task. Think about the task of getting to school. At the beginning of the task you are at home. At the end of the task you are at school. The process description

Appendix J Children's Toys and Games survey

The questionnaire gave the children a list of toys and games and they indicated how much they enjoyed these on a scale of 1-5. The survey was based on the instrument used by Cutts, Q., Patitsas, E., et al., 2018 in their study on early developmental experiences and computing proficiency. Below is an extract from the survey.

Please show how much you enjoyed the following types of toys/game/activities when you were younger. (no experience, tick N/A) None, or low ,Some, Moderate, High, Very high

- Climbing
- Card games
- Making or building models
- Riding bikes
- Playing with balls
- Sand play
- Computer games
- Trampolines
- Wrestling
- Water play
- Playing with stones, sticks or mud
- Painting
- Lego
- Reading
- Games with rules
- Drawing or painting
- Being read to by others
- Singing
- Making music or sounds
- Dolls
- Superheroes
- Dressing up
- Playing at pretend shops
- Play dough
- Playing at schools
- Board games
- Electronic games
- Writing
- Team games
- Guessing games
- Playground games
- Pretend play
- Chasing friends
- Physical play
- Playing with objects
- Dancing
- Playing at pretend houses
- Other (please specify)

Appendix J Sample table of results for P4 School B

ID	SUCCESS CRITERIA SCORING ACROSS THREE LESSONS											
	SC1.1	SC1.2	SC2.1	SC2.2	SC2.3	SC2.4	SC3.1	SC3.2	SC3.3	Total		
41B	A	G	G	G	G	G	G	R	G	24		
42B	A	G	G	G	A	A	A	A	A	21		
43B	A	G	A	A	G	G	A	A	A	22		
44B	A	A	A	A	G	A	A	G	A	20		
45B	A	G	G	G	G	G	G	G	G	26		
46B	A	G	G	G	G	A	G	G	G	25		
47B	A	G	A	G	A	G	G	A	G	23		
48B	A	G	G	A	A	G	A	G	A	22		
49B	A	G	G	A	A	G	G	A	G	23		
410B	A	G	G	G	G	G	G	A	A	24		
411B	A	G	G	G	G	G	A	A	G	24		
412B	A	G	A	G	A	G	A	A	A	22		
413B	A	G	G	A	A	G	A	A	A	21		
414B	A	G	G	G	G	G	G	G	G	26		
415B	A	G	G	G	A	A	G	G	A	23		
416B	A	G	G	R	R	G	G	A	G	21		
417B	A	G	G	A	A	G	A	A	A	21		
418B	R	G	G	G	G	G	R	R	R	19*		
419B	A	G	G	G	G	A	A	G	R	22*		
		%		%		%		%				
R	1	5.2	0	0	0	1	1	0	1	2	2	8
A	18	94	1	5.2	4	6	8	5	9	10	9	
G	0	0	18	94	14	12	10	14	10	7	8	

Appendix L School K Children's view of level of difficulty during the planned learning

▾ Lesson 1: Persistence: Building a Foundation Unplugged Activity	▾ Lesson 11: Events: The Big Event Unplugged Activity
▾ Lesson 2: Programming in Maze 1 2 3 4 5 6 7 8 9 10 11	▾ Lesson 12: Build a Flappy Game 1 2 3 4 5 6 7 8 9 10 11
▾ Lesson 3: Debugging in Maze 1 2 3 4 5 6 7 8 9 10	▾ Lesson 13: Events in Play Lab 1 2 3 4 5 6 7 8 9 10 11
▾ Lesson 4: Real-life Algorithms: Paper Planes Unplugged Activity	▾ Lesson 14: Digital Citizenship: Screen Out the Mean Unplugged Activity
▾ Lesson 5: Programming in Collector 1 2 3 4 5 6 7 8 9 10 11 12 13	▾ Lesson 15: Beyond Programming: Binary Bracelets Unplugged Activity

▾ Lesson 6: Programming in Artist 1 2 3 4 5 6 7 8 9 10
▾ Lesson 7: Loops: Getting Loopy Unplugged Activity
▾ Lesson 8: Loops with Rey and BB-8 1 2 3 4 5 6 7 8 9 10 11 12
▾ Lesson 9: Loops in Artist 1 2 3 4 5 6 7 8 9 10 11 12
▾ Lesson 10: Loops in Harvester 1 2 3 4 5 6 7 8 9 10 11 12

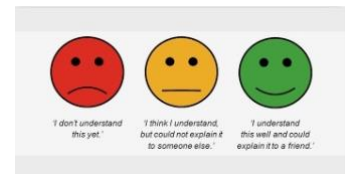
Using traffic lights, how did you find the lesson?

Lesson ____

Activity ____

Lesson

Activity



Appendix M CfE Computing Science 3 Step overview.

(3) APPLICATION OF LEARNING BY CREATING SOLUTIONS



(2) HOW TOOLS AND LANGUAGES USE THE CONCEPTS



(1) CORE COMPUTING SCIENCE CONCEPTS

Glossary

CfE Curriculum for Excellence

CS education Computing science education

Ambassadors: CS4 Final year computing science undergraduate students

AiFL Assessment for Learning strategies

Teacher/Educator Adult leading the learning in the class

Theoretical content studies

¹ Patton, M.Q., 1987. *How to use qualitative methods in evaluation* (No. 4). Sage.