



**PROGRAMMING LANGUAGES, CURRICULUM AND COMPUTATIONAL
THINKING AT A COGNITIVE LEVEL OF FORMAL OPERATIONS**

by

WILHELM COENRAAD ROTHMAN

Thesis submitted in fulfilment of the requirements for the degree

Doctor of Technology: Information Technology

in the Faculty of Informatics and Design

at the Cape Peninsula University of Technology

Supervisor: Dr André de la Harpe

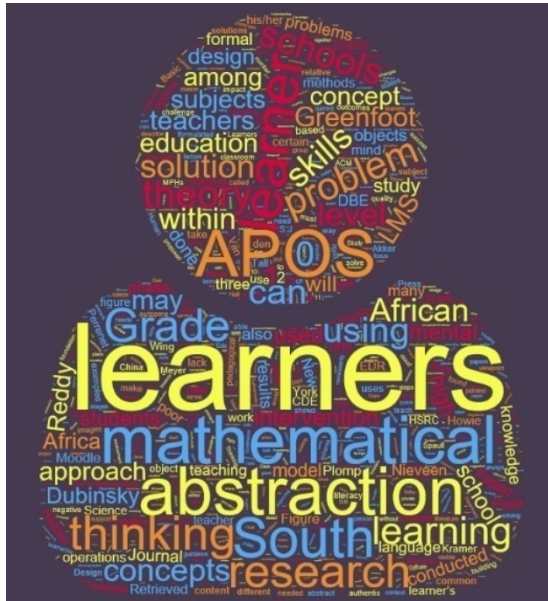
Co-supervisor: Prof Johannes Cronje

District Six

Date submitted: 31 August 2020

CPUT copyright information

The thesis may not be published either in part (in scholarly, scientific or technical journals), or as a whole (as a monograph), unless permission has been obtained from the University.



DECLARATION

I, Wilhelm Coenraad Rothman, declare that the contents of this thesis represent my own unaided work, and that the thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

Signed

31 August 2020

Date

PROOFREADING CERTIFICATE

15 September 2020

WILHELM COENRAAD ROTHMAN

Faculty of Informatics and Design
Cape Peninsula University of Technology
Cape Town

RE: CERTIFICATE - TECHNICAL EDITING AND PROOFREADING OF DOCTORAL THESIS

I, the undersigned, herewith certify that the technical editing and proofreading of the Doctoral thesis of Wilhelm Coenraad Rothman, entitled "*Programming Languages, Curriculum, and Computational Thinking at a Cognitive Level of Formal Operations*", has been conducted and concluded.

The finalised thesis was submitted to Mr Rothman on 15 September 2020 and cc'd to Prof Andre de la Harpe.

Sincerely



Professor Annelie Jordaan
DTech: Information Technology
Ph: 065 990 3713

Member: SATI 1003347



South African Translators' Institute (SATI)

TURNITIN REPORT

PROGRAMMING LANGUAGES, CURRICULUM AND COMPUTATIONAL THINKING AT A COGNITIVE LEVEL OF FORMAL OPERATION

ORIGINALITY REPORT

5 %	%	%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	creativecommons.org Internet Source	1 %
2	hdl.handle.net Internet Source	<1 %
3	link.springer.com Internet Source	<1 %

ABSTRACT

High school learners underperform at a cognitive level of formal operations when engaging in subjects such as Mathematics and Science. Computational thinking is concerned with abstract methodology supporting mathematical thinking. The problem statement of this research states that it is unclear how computational thinking can be enhanced among high school learners at a cognitive level of formal operations. This “wicked” problem was investigated by asking two research questions, namely: i) “What are the characteristics of an enhanced learner’s teaching and learning strategy that can empower learners to master computational thinking skills through APOS theory, infused by a programming language at high school level?” and ii) “How can computational thinking skills at a cognitive level of formal operations be promoted among high school learners through the teaching of a programming language aligned to Action Process Object Schema (APOS)?”

The aim of this research was to explore and understand how a programming language, using APOS theory as lens, could promote computational thinking skills at a cognitive level of formal operations among high school learners. The study was conducted at a private high school in the Western Cape.

The research methodology was based on an interpretivist research philosophy. The ontological underpinning of the study was subjective and the epistemological stance accepted opinions of learners through written, spoken and visual attributed meanings. The axiology of the researcher was that of a practising educator in programming, a teaching and learning expert and a certified Java-Greenfoot instructor through Oracle.

Data were collected during lectures, observations, interviews and assignments. Using Greenfoot as a programming language, supported by Moodle as LMS, learners discovered programming through “worked examples”. Qualitative data analysis was done through data condensation, data display, and drawing and verification of conclusions using thematic analysis. Ethical considerations were enforced by the ethical standards of the university of study, maintaining a high level of confidentiality towards all subjects at all times. The research strategy was based on Educational Design Research (EDR) as a validation study through interventions. Findings show that computational thinking can be promoted among learners at a cognitive level of formal operations through Greenfoot programming language with APOS theory as lens.

Keywords: APOS theory, cognitive level of formal operations (CLFO), computational thinking, embodied cognition, educational design research (EDR), Learner Management System (LMS), programming language

ACKNOWLEDGEMENTS

I wish to thank:

- Dr André de la Harpe, my supervisor, for his guidance, encouragement and support in completing this research
- Prof Johannes Cronjé, my co-supervisor for his advice, wisdom and contagious enthusiasm
- The organisations and individuals that participated in this research for their support and time to share their insights
- My wife Melissa, daughters Grethe, Margot and son Loki for their love and unconditional support
- To my mother, for always believing in me, but who is not able to comprehend life anymore
- My family and friends for their emotional support, understanding and encouragement in following my academic aspirations
- Curro Private School Durbanville, Durbanville High School and Bosmansdam High School, who welcomed my efforts and took me in as part of the staff complement

The financial assistance of the National Research Foundation towards this research is acknowledged. Opinions expressed in this thesis and the conclusions arrived at, are those of the author, and are not necessarily to be attributed to the National Research Foundation.

TABLE OF CONTENTS

DECLARATION	ii
PROOFREADING CERTIFICATE	iii
TURNITIN REPORT	iv
ABSTRACT	v
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	xvii
LIST OF TABLES	xxi
GLOSSARY/ACRONYMS	xxii
TERMS AND DEFINITIONS	xxiii
CHAPTER 1: INTRODUCTION	1
1.1 Introduction	2
1.2 Computational thinking and the mathematics connection	3
1.3 Computational thinking and the programming connection	3
1.4 Rationale of the research	4
1.5 Research problem statement.....	5
1.6 Aim of the research	5
1.6.1 Objectives of the research.....	5
1.7 Research questions (RQs)	6
1.8 Research focus	7
1.9 Methodological considerations	8
1.9.1 Research paradigm and research philosophy.....	8
1.9.2 Research approach	9
1.10 Research design	10
1.10.1 Sampling strategies.....	10
1.10.2 Data collection strategies	11
1.11 Contribution of the research	11
1.11.1 Theoretical contribution	11
1.11.2 Contribution to academic discipline	11
1.11.3 Methodological contribution.....	12
1.11.4 Practical contribution	12
1.12 Ethical considerations	12
1.13 Assumptions.....	13
1.14 Delineation of the research.....	13
1.15 Conclusion	13

1.16	Summary.....	14
1.17	Structure of the thesis	15
CHAPTER 2: LITERATURE REVIEW		16
2.1	Introduction and background	16
2.2	Literature review.....	17
2.2.1	Research problem	18
2.2.2	Search and acquisition process.....	19
2.2.2.1	Research question (RQ) 1	22
2.2.2.2	Research question (RQ) 2.....	58
2.2.3	Theoretical conceptual framework.....	76
2.2.4	The viewpoint of educators and professionals on programming	81
2.2.5	Target group.....	84
2.3	Literature review summary	85
2.3.1	Mathematics research	88
2.3.2	How does one involve the whole body?.....	89
2.3.3	Discovery learning creates challenges	90
2.3.4	Status quo of teaching and learning	91
CHAPTER 3: DESIGN RESEARCH		93
3.1	Introduction	93
3.2	Design Research (DR)	93
3.2.1	Design Science Research (DSR)	94
3.2.2	Design-Based Research (DBR).....	95
3.2.3	Educational Design Research (EDR).....	96
3.2.3.1	Introduction	96
3.2.3.2	Paradigms in EDR.....	96
3.2.3.3	EDR approaches.....	97
3.2.3.4	EDR as model of choice	99
3.3	EDR implementation	100
3.3.1	Introduction	100
3.3.2	Step 1: Identification of the main phases of EDR.....	101
3.3.3	Step 2: Simplify the wicked problem.....	101
3.3.3.1	EDR research question	102
3.3.4	Step 3: The general phases of EDR	104
3.3.4.1	Phase 1: Preliminary research.....	104
3.3.4.2	Phase 2: Prototyping	105
3.3.4.3	Phase 3: Assessment.....	105

3.4	Summary.....	106
CHAPTER 4: RESEARCH DESIGN		107
4.1	Introduction	107
4.2	Research paradigms	108
4.3	Research philosophy.....	110
4.3.1	Ontology – The nature of reality	111
4.3.2	Epistemology – The nature of knowledge.....	111
4.3.3	Axiology – The role of values and ethics	112
4.4	Research approach.....	112
4.5	Research strategy	113
4.5.1	The demonstration case	113
4.5.1.1	Background.....	113
4.5.1.2	The school visiting and planning process	113
4.5.2	Action research (AR)	115
4.5.3	Design Research (DR) strategy.....	117
4.5.4	Intervention development	118
4.5.4.1	Introduction	118
4.5.4.2	Intervention 1: Abstraction (abstract thinking) assessment (Appendix B-1).....	118
4.5.4.3	Intervention 2: Implement Greenfoot programming language (Appendix C).....	121
4.5.4.4	Intervention 2A: Introduction of a Genetic Decomposition process (Adapted from Arnon et al., 2014:112; Appendices D-1, D-2).....	122
4.5.4.5	Intervention 2B: Introduction of an enhanced Genetic Decomposition of “Load a Greenfoot Scenario” (Appendix D-2).....	123
4.5.4.6	Intervention 2C: Help documentation in Greenfoot (Appendix D-3)	123
4.5.4.7	Intervention 3: Interaction with the Moodle LMS (Appendix E-1).....	124
4.5.4.8	Sub-Intervention 3A: Introduction of the Moodle LMS (Appendix E-1)	124
4.5.4.9	Intervention 3B: Juggling enactment to enforce Moodle usage among learners (Appendix E-2).....	124
4.5.4.10	Intervention 3C: Moodle and generalised terminology (Appendix 3C).....	125
4.5.4.11	Intervention 4: Creating a Moodle Learner Management System (LMS).....	125
4.5.4.12	Intervention 4A: Creating a Linux Server with external access (Appendix F-1) ..	126
4.5.4.13	Intervention 4B: Creating a cloud-based Moodle LMS (Appendix F-2).....	126
4.5.4.14	Intervention 5: Greenfoot access	127
4.5.4.15	Intervention 5A: Introduction to Greenfoot (Appendix G-1)	127
4.5.4.16	Intervention 5B: Revisit previous activities (Appendix G-2).....	127
4.5.4.17	Intervention 6: Applying Process and Object within mathematics (Appendix H).	128
4.5.4.18	Intervention 7: Greenfoot as Process and Object (Appendix I).....	129

4.5.4.19	Intervention 8: Rollout of code in Greenfoot (Appendix J)	129
4.5.4.20	Intervention 9: Making decisions towards Encapsulation (Appendix K).....	130
4.5.4.21	Intervention 10: Revisit encapsulation with Randomize option (Appendix L).....	130
4.5.4.22	Intervention 11: Assessment (Appendix M).....	131
4.5.4.23	Intervention 11A: Informing learners of the assessment in a structured manner (Appendix M-1)	131
4.5.4.24	Intervention 11B: Assessment in Greenfoot on Encapsulation and problem solving (Appendix M-2)	131
4.5.4.25	Intervention 12: The variable in Greenfoot (Appendix N)	132
4.5.4.26	Intervention 13: Moving from Process to Object in APOS using Greenfoot (Appendix O)	133
4.5.4.27	Intervention 14: GD creation on IF statement	134
4.5.4.28	Intervention 14A: Basic creation of scenario with World and Actor classes (Appendix P-1).....	134
4.5.4.29	Intervention 14B: Manipulation of Actors in a World (Appendix P-2)	134
4.5.4.30	Intervention 14C: Interaction of Actor within the World solving problems (IF statement as precursor to GD) (Appendix P-3)	135
4.5.4.31	Intervention 14D: Adding graph paper as part of GD to develop algorithm (Appendix P-4).....	136
4.5.4.32	Intervention 14E: The IF statement as a solution to address problems (Appendix P-5)	136
4.5.4.33	Intervention 15: Testing Greenfoot to be accepted among teachers (Appendix Q)	137
4.5.4.34	Intervention 16: Creating an arcade game (Appendix U)	137
4.5.5	Interviews	138
4.5.5.1	Interviews IA and IB: Algebra exercise on simplification; Science assessment question and Voltage-Ampere-Resistance pyramid (Appendices R-1, R-2, R-3 and R-4) ..	138
4.5.6	Observations	138
4.5.7	Data collection strategies	139
4.5.8	Sampling	139
4.5.9	Data analysis.....	140
4.6	Summary.....	141
CHAPTER 5: DATA ANALYSIS AND FINDINGS.....		142
5.1	Introduction	143
5.2	Mathematical belief system	143
5.3	EDR phases.....	147
5.4	Data collection and analysis	147

5.4.1	Introduction	147
5.4.2	Phase 1: Preliminary research phase	147
5.4.2.1	Needs and context analysis.....	148
5.4.2.2	The literature review	149
5.4.2.3	Theory development.....	154
5.4.2.4	Target group.....	155
5.4.3	Phase 2: Prototyping/ Enactment phase.....	155
5.4.3.1	Introduction	155
5.4.3.2	Intervention 1: Abstraction (Abstract Thinking) assessment (Appendix B-1)	160
5.4.3.3	Intervention 2: Implement the Greenfoot programming language (Appendix C-1).....	162
5.4.3.4	Intervention 2A: Introduction of a Genetic Decomposition (GD) (Appendices D-1, D-2)	171
5.4.3.5	Intervention 2B: Introduction of an enhanced GD (Appendix D-1)	173
5.4.3.6	Intervention 2C: Help documentation in Greenfoot (Appendix D-3)	177
5.4.3.7	Intervention 3: Interaction with the Moodle LMS (Appendix E-1).....	177
5.4.3.8	Intervention 3A: Tools for Moodle (Appendix E-1)	178
5.4.3.9	Intervention 3B: Juggling enactment to enforce Moodle usage among learners (Appendix E-2).....	180
5.4.3.10	Intervention 3C: Moodle and generalised terminology (Appendix E-3).....	184
5.4.3.11	Intervention 4: Creating a Moodle Learner Management System (LMS).....	189
5.4.3.12	Intervention 4A: Creating a Linux Server with external access (Appendix F-1) ..	189
5.4.3.13	Intervention 4B: Creating a cloud-based Moodle LMS (Appendix F-2).....	191
5.4.3.14	Intervention 5: Greenfoot Access.....	193
5.4.3.15	Intervention 5A: Greenfoot access (Appendix G-1).....	194
5.4.3.16	Intervention 5B: Revisit previous activities (Appendix G-2)	198
5.4.3.17	Intervention 6: Applying Process and Object within mathematics (Appendix H) .	199
5.4.3.18	Intervention 7: Greenfoot as process and object (Appendix I).....	200
5.4.3.19	Intervention 8: Rollout of code in Greenfoot (Appendix J).....	203
5.4.3.20	Intervention 9: Making decisions towards Encapsulation (Appendix K).....	207
5.4.3.21	Intervention 10: Revisit encapsulation with the Randomize option (Appendix L)	212
5.4.3.22	Intervention 11: Assessment (Appendix M).....	214
5.4.3.23	Intervention 11A: Informing the learners of the assessment in a structured manner (Appendix M-1)	214
5.4.3.24	Intervention 11B: Assessment in Greenfoot on Encapsulation and problem solving (Appendix M-2)	215
5.4.3.25	Intervention 12: The Variable in Greenfoot (Appendix N).....	217

5.4.3.26	Intervention 13: Moving from Process to Object in APOS using Greenfoot (Appendix O)	219
5.4.3.27	Intervention 14: GD creation on IF statement	222
5.4.3.28	Intervention 14A: Basic understanding of a scenario with World and Actor classes (Appendix P-1).....	223
5.4.3.29	Intervention 14B: Manipulation of Actors in a World (Appendix P-2)	225
5.4.3.30	Intervention 14C: Interaction of Actor within the world solving problems (IF statement as precursor to GD) (Appendix P-3)	227
5.4.3.31	Intervention 14D: Adding graph paper as part of GD to develop algorithm (Appendix P-4).....	231
5.4.3.32	Intervention 14E: The IF statement as a solution to address problems (Appendix P-5)	233
5.4.3.33	Intervention 15: Testing Greenfoot to be accepted among teachers (Appendix Q)	234
5.4.3.34	Intervention 16: Creating an Arcade Game (Appendix U)	236
5.4.4	Interviews	237
5.4.4.1	Interviews: Algebraic Simplification and the Electrical Circuit Diagram (Appendix R)	237
5.4.4.2	Interview IA: Algebra Exercise on Simplification (Appendix R-1)	238
5.4.4.3	Interview 1B: Electrical Circuit (Appendix R-2 and R-3).....	240
5.4.5	Phase 3: Assessment.....	243
5.4.5.1	Themes within interventions	243
5.4.5.2	Themes within interviews	246
5.5	Summary.....	246
CHAPTER 6: DISCUSSION.....		248
6.1	Introduction	248
6.2	Themes	249
6.2.1	Interventions.....	249
6.2.1.1	APOS theme	249
6.2.1.2	Beliefs theme	254
6.2.1.3	Cognitive Balance theme.....	255
6.2.1.4	Computational Thinking Theme.....	257
6.2.1.5	Learning theme	259
6.2.1.6	Learner Management System theme.....	259
6.2.1.7	Programming language theme	261
6.2.1.8	Technical theme.....	263
6.2.2	Interviews	264

6.2.2.1	APOS	265
6.2.2.2	Beliefs	266
6.2.2.3	Cognitive Balance	266
6.2.2.4	Computational Thinking.....	266
6.2.2.5	Learning	266
6.3	Summary.....	267
CHAPTER 7: CONCLUSION AND RECOMMENDATIONS		269
7.1	Introduction	269
7.2	Conclusions.....	270
7.2.1	Linking RQ 1 with the findings	271
7.2.2	Linking RQ 2 with the findings	273
7.2.3	EDR Question	276
7.3	Overview of the Study	277
7.4	Research Chapters	278
7.4.1	Chapter 2 – Literature review	278
7.4.2	Chapter 3 – Design Research	279
7.4.3	Chapter 4 – Research Design	280
7.4.4	Chapter 5 – Data analysis and findings	280
7.4.5	Chapter 6 – Discussion	281
7.5	Summary.....	281
7.5.1	Conclusions.....	281
7.5.2	Recommendations	281
7.5.2.1	Programming language.....	281
7.5.2.2	Technical.....	282
7.5.2.3	Cognitive Balance and Beliefs	282
7.5.2.4	LMS.....	282
7.5.2.5	Learning	283
7.5.2.6	APOS	283
7.5.2.7	Education	283
CHAPTER 8: CONTRIBUTION, FURTHER RESEARCH AND REFLECTIONS		285
8.1	Introduction	285
8.2	Contributions of the research.....	286
8.2.1	Theoretical contribution	286
8.2.2	Contribution to academic discipline	287
8.2.3	Methodological contribution.....	287
8.2.4	Practical contribution	288

8.3	Further Research	289
8.4	Reflection	289
8.4.1	Learning perspective	289
8.4.2	Research perspective.....	290
8.5	Assessment of research	292
8.6	Assessment of the context and research purpose	292
8.7	Self-reflection	292
REFERENCES		295
APPENDICES		318
Appendix A: Introductory letters for the collection of research data.....		318
Appendix A-1: Curro Private School		318
Appendix A-2: Ethics Clearance from CPUT.....		319
Appendix A-3: Chester House Private School		320
Appendix A-4: Bosmansdam High Public School		321
Appendix A-5: Letter of Introduction to Schools.....		322
Appendix A-6: Videos Uploaded on Moodle Site for Learners to Access.....		323
Appendix A-7: Greenfoot Developer.....		324
Appendix A-8.1.1: Teachers Rollout Course on Greenfoot.....		325
Appendix A-8.1.2: Teachers Rollout Course on Greenfoot.....		326
Appendix A-8.2.1: Rollout to WCED Teachers Workshop 1.....		327
Appendix A-8.2.2: Rollout to WCED Teachers.....		328
Appendix A-8.2.3: Location sent to Teachers.....		329
Appendix A-8.2.4: Timesheet o.b.o. Oracle for WCED Teachers Training Workshop 1		330
Appendix A-8.3.1: Rollout to WCED Teachers Workshop 2.....		331
Appendix A-8.3.3: Location of WCED Teachers Workshop 2		332
Appendix A-8.3.4: Timesheet o.b.o. Oracle for WCED Teachers Training Workshop 2		333
Appendix A-8.3.5: WCED Teachers Training Signatures Workshop 2		334
Appendix A-8: Oracle Instructors Certificate.....		334
Appendix A-9: Emails sent to Authors of the APOS Theory.....		335
Appendix A-9.1: Email 1 to Prof Dubinsky (12 February 2015).....		335
Appendix A-9.2: Email 2 from Prof Dubinsky (15 March 2015).....		335
Appendix A-9.3: Email 2 from Prof Dubinsky (21 March 2015).....		336
Appendix A-9.4: Email to Dr Illana Arnon		337
Appendix A-9.5.1: Email Response from Dr Arnon (8 March 2015).....		338

Appendix A-9.5.2: Email Response from Dr Arnon in attached Word document 04-12-2015 @ 8:44 pm.....	338
Appendix A-9.6: Western Cape Education Plan to Salvage Mathematics.....	339
Appendix B-1: Intervention 1 (Abstraction [Abstract Thinking] Assessment)	340
Appendix B-2: Questionnaire on Mathematics in General (Learner X)	341
Appendix B-3: Questionnaire on Mathematics in General (Learner Y)	342
Appendix B-4: Questionnaire on Mathematics in General (Learner Z)	343
Appendix C: Intervention 2A-1 computational thinking in motion (Compiled interpretation taken from Denning (2017) and AHO(2011)).....	344
Appendix D-1: Intervention 2A-2 (Genetic Decomposition Process adapted from Arnon et al., 2004)	345
Appendix D-2: Intervention 2A-3 (Genetic Decomposition of “Load a Greenfoot Scenario”)	346
Appendix D-3: Intervention 2B (Help Documentation in Greenfoot)	347
Appendix D-4: Intervention 14E (Genetic Decomposition of IF statement)	348
Appendix E-1: Intervention 3A (Introduction of the Moodle LMS).....	349
Appendix E-2: Intervention 3B (Juggling enactment to enforce Moodle usage among learners)	350
Appendix E-3: Intervention 3C (Moodle and Generalised Terminology).....	351
Appendix F-1: Intervention 4A (Creating a Linux Server with external access).....	352
Appendix F-2: Intervention 4B (Creating a Cloud-based Moodle LMS)	352
Appendix G-1: Intervention 5A (Introduction to Greenfoot)	353
Appendix G-2: Intervention 5B (Revisit previous Activities)	354
Appendix H: Intervention 6 (Applying Process and Object within mathematics).....	355
Appendix I: Intervention 7 (Greenfoot as Process and Object).....	356
Appendix J Intervention 8: Rollout of code in Greenfoot in Figure 4.20	357
Appendix K: Intervention 9 (Making decisions towards Encapsulation).....	358
Appendix L: Intervention 10 (Revisit encapsulation with Randomize option)	359
Appendix M-1: Intervention 11A (Informing the learners of the assessment in a structured manner).....	360
Appendix M-2: Intervention 11B (Assessment in Greenfoot on Encapsulation and problem solving).....	361
Appendix N: Intervention 12 (The Variable in Greenfoot).....	362
Appendix O: Intervention 13 (Moving from Process to Object in APOS using Greenfoot)	364
Appendix P-1: Intervention 14A: Basic creation of scenario with World and Actor classes.....	365
Appendix P-2: Intervention 14B: Manipulation of Actors in a World.....	366

Appendix P-3: Intervention 14C: Interaction of Actor within the world solving problems	367
Appendix P-4: Intervention 14D: APOS theory as tool to investigate problem questions	368
Appendix P-5: Intervention 14E: The IF statement as a solution to address problems	369
Appendix Q: Rollout to WCED Schools	370
Appendix R-1: Task for interview 1A (Algebra Exercise on Simplification)	371
Appendix R-2: Task for interview 1B (Science Assessment Question)	371
Appendix R-3: Interview: Voltage-Ampere-Resistance pyramid	372
Appendix R-4: Interviews on Mathematics and Science	372
Appendix S: Greenfoot Name Badge	379
Appendix T: Order Form for the Greenfoot Badges	380
Appendix U: First Game Development	381
Appendix V: Putting it Together	384
Appendix W: Table of interventions and Actions	386
Appendix X: FEDS for Artefact Evaluation	388
Appendix Y: Themes	389

LIST OF FIGURES

Figure 1.1: Chapter 1 layout.....	1
Figure 1.2: A 25-step model in Neo-Piagetian cognitive development and Neo-Eriksonian social-affective development (Adopted from Young, 2012:242)	7
Figure 1.3: Research Onion (Adopted from Saunders, Lewis & Thornhill, 2019:130).....	9
Figure 2.1: Chapter 2 layout.....	16
Figure 2.2: Hermeneutic framework for the literature review process (Adopted from Boell & Cecez-Kecmanovic, 2014:264)	18
Figure 2.3: Scopus Database search outcome on “Abstraction in Mathematics”	20
Figure 2.4: Bruner’s three modes of representation (Adopted from Tall, 2003:2)	25
Figure 2.5: Three representational worlds and their associations with other viewpoints (Adopted from Tall, 2003:4)	25
Figure 2.6: Three worlds in mathematics (Adopted from Tall, 2008:4)	27
Figure 2.7: Translate the “Appleness” of counting into a number object (Adapted from Meyer, 2010:2)	33
Figure 2.8: Model of predictors of OOP performance (Adopted from Cegielski & Hall, 2006:74)	36
Figure 2.9: Learning-by-design cycles (Adopted from Lee & Kolodner, 2011:6).....	37
Figure 2.10: Computational thinking in motion (Adapted from Denning, 2017; Aho, 2011) ...	38
Figure 2.11: Cognitive development style of PLs (Adopted from White & Sivitanides, 2002:63)	39
Figure 2.12: Procedural and conceptual knowledge (Adopted from Tall, 2008:12).....	46
Figure 2.13: Tearing corners of a triangle to form a straight line (Adopted from Tall, 2002:9)	46
Figure 2.14: A learning framework (Adopted from Tall, 2008:14)	48
Figure 2.15: An Actor class using a Stride scenario (Greenfoot ver 3.0)	54
Figure 2.16: Actor Camel class within a Java scenario (Greenfoot 3.0)	55
Figure 2.17: Schema and its construction (Adopted from Dubinsky, 1991:106).....	58
Figure 2.18: Diagram based on Grossman’s Reformulation of PCK (Adopted from Saeli et al., 2011:76)	61
Figure 2.19: TPACK framework accommodating technology (Adopted from Koehler & Mishra, 2009:63)	63
Figure 2.20: Interpretation and adaptation of APOS theory from a research and curriculum development stance (Adapted from Arnon et al., 2014:112).....	70
Figure 2.21: The theoretical conceptual framework for the improvement of computational thinking among learners.....	76
Figure 3.1: Layout of Chapter 3	93

Figure 3.2: Framework and Context of DR (Adopted from Venable, 2006:3).....	94
Figure 3.3: DSR process model (Adopted from Vaishnavi, Kuechler & Petter, 2019:14)	95
Figure 3.4: Perceptions of EDR objectives and methods (Adopted from Weber, 2010:4).....	96
Figure 3.5: Refinement of problems, solutions, methods and design principles (Adopted from Reeves, 2006:14)	98
Figure 3.6: Generic model for EDR (Adopted from McKenney & Reeves, 2012:14).....	98
Figure 3.7: Questions and methods for DR using ILDF (Adopted from Bannan, 2013:55)....	99
Figure 3.8: Synthesised generic model for educational DR (Adopted from Van Wyk & De Villiers, 2018:305)	101
Figure 3.9: Overview of phases (Adopted from Plomp, 2013:19)	104
Figure 3.10: Diagram showing the problem identification phase (Adopted from Plomp, 2013:19)	104
Figure 4.1: Layout of Chapter 4	107
Figure 4.2: Four quadrants of sociological and organisational research (Adopted from Burrell & Morgan, 1979:22)	109
Figure 4.3: Research Onion (Adopted from Saunders, Lewis & Thornhill, 2019:130).....	110
Figure 4.4: One term's visits to the private school.....	114
Figure 4.5: Time table structured on a two-week period for first term.....	114
Figure 4.6: Time table structured on a two-week period for second term	115
Figure 4.7: Five phases of Action Research Method (Adopted from Susman & Evered, 1978 as illustrated by Järvinen, 2007:39).....	115
Figure 4.8: Composite representation of DR, DSR and DBR (Adopted from Van Wyk & De Villiers, 2018:304)	117
Figure 4.9: Relationship among DS, DR, DSR and EDR (Adapted from Venable, Pries-Heje & Baskerville, 2016:141; Miah, Solomonides & Gammack, 2010:2).....	118
Figure 4.10: Abstraction exercise.....	119
Figure 4.11: Computational thinking in motion (Adapted from Denning, 2017; Aho, 2011) .	121
Figure 5.1: Chapter Layout	142
Figure 5.2: Word Cloud on this thesis	142
Figure 5.3: Learner X answers questionnaire on geometry knowledge (Appendix B-2).....	145
Figure 5.4: Learner Y answers questionnaire on geometry knowledge (Appendix B-3).....	145
Figure 5.5: Learner Z sees Mathematics as an obstacle Appendix (B-4).....	146
Figure 5.6: Cross Section of EDR model (Adopted from Van Wyk & De Villiers, 2018:305)	147
Figure 5.7: Cross section of EDR model (Adopted from Van Wyk & De Villiers, 2018:305)	153
Figure 5.8: Cross section of EDR model (Adopted from Van Wyk & De Villiers, 2018:305)	154
Figure 5.9: The theoretical conceptual framework for learners.....	155
Figure 5.10: Cross section of EDR model (Adopted from Van Wyk & De Villiers, 2018:305)	156

Figure 5.11: Depicts the flow of Phase 2.....	156
Figure 5.12: Adaptation of the theoretical conceptual framework to enhance computational thinking among learners in Grade 8	164
Figure 5.13: The Wombat Actor object within MyWorld World Object	168
Figure 5.14: The proposed theoretical conceptual framework for enhanced learning using programming language and LMS.....	175
Figure 5.15: Questionnaire example	181
Figure 5.16: Comments on Juggling	182
Figure 5.17: General terminology.....	185
Figure 5.18:General terminology.....	185
Figure 5.19: General terminology.....	185
Figure 5.20: General terminology.....	185
Figure 5.21:General terminology.....	185
Figure 5.22: General terminology.....	186
Figure 5.23: General terminology.....	186
Figure 5.24: General terminology.....	186
Figure 5.25: General terminology.....	187
Figure 5.26: General terminology.....	187
Figure 5.27: General terminology.....	187
Figure 5.28: General terminology.....	188
Figure 5.29: (Adopted from https://www.filehippos.org/java-development-kit/)	194
Figure 5.30: Greenfoot editor with World Code	196
Figure 5.31: Greenfoot “SpaceWorld” scenario	196
Figure 5.32: Greenfoot “SpaceWorld” World scenario populated by Rocket objects	197
Figure 5.33: The Wombat Actor object within MyWorld World Object	201
Figure 5.34: The Wombat Actor code in Greenfoot editor	202
Figure 5.35: Greenfoot Class Documentation	204
Figure 5.36: Greenfoot ChessBoard World	205
Figure 5.37: Chessboard when using (600, 400, 1) dimension.....	205
Figure 5.38: Learner calculating the size of the World that will produce a perfect chess board	206
Figure 5.39: A learner’s response on Activity question.....	209
Figure 5.40: TurtleWorld in Greenfoot.....	209
Figure 5.41: The Turtle act () method coding	210
Figure 5.42: Encapsulating code.....	210
Figure 5.43: Greenfoot code illustrating the “turn” command.....	213
Figure 5.44: Greenfoot code for repeating code.....	216
Figure 5.45: Question 7 answer	224

Figure 5.46: Question 3 on the term 'compilation'	226
Figure 5.47: Answer to what 'compilation' stand for	226
Figure 5.48: Answer consists of Greenfoot code for checking edges	227
Figure 5.49: Inspection parameters on Turtle object	228
Figure 5.50: Answer to Question 1	228
Figure 5.51: Response from student regarding Car movement at edge of the World	229
Figure 6.1: Chapter 6 Layout	248
Figure 6.2: Flow of APOS theme.....	249
Figure 6.3: Computational thinking hub.....	257
Figure 7.1: Chapter layout.....	269
Figure 8.1: Chapter 8 Layout	285
Figure 8.2: Reflection of this EDR research (Adopted from van Wyk & De Villiers, 2018:305)	290
Figure 8.3: Processes flow diagram of this research (Adapted from Van Wyk & De Villiers, 2018:306)	291
Figure 8.4: Proposed conceptual framework.....	291

LIST OF TABLES

Table 1.1: Mapping of research methods and objectives to RQ 1	6
Table 1.2: Mapping of research methods and objectives to RQ 2	7
Table 1.3: Structure of the thesis	15
Table 2.1: Genetic Decomposition	72
Table 3.1: The structure of the EDR question	103
Table 4.1: Abstract thinking among grade 8 learners	120
Table 5.1: Mapping of research methods and goals to Research Question 1	152
Table 5.2: Mapping of research methods to Research Question 2	152
Table 5.3: The research EDR goals and objectives	153
Table 5.4: Criteria for high quality Interventions	157
Table 5.5: Detailed summary of interventions	158
Table 5.6: Abstract thinking among grade 8 learners	161
Table 5.7: The prerequisites for using Moodle	179
Table 5.8: The results of Intervention 3B is	181
Table 5.9: Moodle csv file structure	192
Table 5.10: Test 2 results	221
Table 5.11: Summary of the findings and themes of the interviews	242
Table 5.12: Themes from Interviews	242
Table 5.13: Grouping of findings, summary of findings, categories and themes	243
Table 5.14: Relationship of research questions, objectives, findings, main findings and themes	245
Table 5.15: Grouping of interviews, summary of findings and themes	246
Table 6.1: Summary of findings per theme	249

GLOSSARY/ACRONYMS

Acronym/Abbreviation	Full Word/Term
AMESA	Association for Mathematics Education in South Africa
ANA	Annual National Assessment
APOS	Action Process Object Schema
AR	Action Research
ASSA	Actuarial Society of South Africa
CAPS	Curriculum Assessment Policy Statements
CLT	Cognitive Load Theory
CLFO	Cognitive Learning of Formal Operations
CS	Computer Science
DBR	Design-Based Research
DR	Design Research
DS	Design Science
DSR	Design Science Research
EDR	Educational Design Research
EEG	Electroencephalogram
GD	Genetic Decomposition
IEA	International Association for the Evaluation of Educational Achievements
ID	Instructional Designer
LMS	Learning Management System
EEG	Electroencephalogram
MBSA	Meta-Belief System Activity
MPH	Mathematical Powerhouse
MPS	Mathematical Problem Solving
NCTM Standards	National Council of Teachers of Mathematics Standards in the USA in 1989
PL	Programming Language
POMI	Path of Minimal Interaction
OOP	Object-Oriented Programming
SAMS	South African Mathematical Society
SAMSAK	South African Mathematics Swiss Army Knife
SCRATCHED	Educators using Scratch at website https://scratched.gse.harvard.edu/
TIMSS	Third International Mathematics and Science Study
TELEs	Technology Enhanced Learning Environments

TERMS AND DEFINITIONS

Terms	Definition/Explanation
Abstractions	Abstractions called computational models are at the heart of computation and computational thinking (Denning, 2017:35).
Algorithm	An Algorithm for a function f is just a Turing machine that computes f (Aho, 2011:5). It is any sequence of steps controlled by a computational model (Denning, 2017:36).
Algorithm design	Algorithm design is a design that controls any machine that uses the computational model to produce a desired effect in the world (Selby & Woollard, 2014). It is expressions of recipes for carrying out tasks; no awareness of computational models is needed (Denning, 2017:37).
Cognitive	The Merriam-Webster dictionary defines cognitive as “Relating to, being, or involving conscious intellectual activity (such as thinking, reasoning, or remembering)” (Merriam-Webster.com, 2019).
Cognitive level of formal operations	The final Piaget’s stage of development is known as the formal operational stage and is present when someone reaches about the age of 12 and continues into adulthood. Deductive way of thinking and understanding conceptual thoughts are present (Cherry, 2014).
Cognitive structures	How concepts within a domain are organised and interrelated within a person’s mind as the building blocks of meaningful learning and retention of instructional materials (Ifenthaler, Masduki & Seel, 2011:41).
Computation	Computation refers to a process that is defined in terms of some underlying model of computation. A model of computation is a mathematical abstraction of a computing system. For a clear definition, the model must be well-defined such as the model of sequential computation in CS or Turing machine. Other models, such as concurrent models define concurrent computation (Aho, 2011:1-6).
Computational thinking	Computational thinking refers to thought processes of abstraction, decomposition, algorithmic design, evaluation and generalisation. However, from an educational perspective it should include the idea of a thought process, the concept of abstraction, and the concept of decomposition (Selby & Woollard, 2014:1,3).
Electroencephalograms (EEG)	An electroencephalogram (EEG) is a test that detects electrical activity in the brain using small, metal discs (electrodes) attached to your scalp (Blocka, 2017:1).
Embodied cognition (EC)	Embodiment is a radical hypothesis that the brain is not the only resource in solving problems. The body of a human being and the perceptions from guided motions assist in solving problems, removing the need of complex internal mental representations to achieve the same goal (Wilson & Golonka, 2013:1).
Epistemological fraud	The pupil produces an exact response, but not because he has “understood and solved the problem”, not because he has learned a mathematical object, but simply because he has established a similarity with another exercise (D’Amore, 2008:11).
Greenfoot	Greenfoot is an integrated educational software development environment aimed at learning and teaching programming to young novices. The target user group starts at pupils from about 14 years of age, and also includes introductory university education (Kölling, 2010b:1).

Terms	Definition/Explanation
High road transfer	High road transfer depends on mindful abstraction from the context of learning or application and a deliberate search for connections: Such a transfer is not reflexive in general. It demands time for exploration and the investment of mental effort. It can easily accomplish far transfer, bridging between contexts (Perkins & Salomon, 1992:8).
Low road transfer	Low road transfer happens when stimulus conditions in the transfer context are sufficiently similar to those in a prior context of learning to trigger well-developed semi-automatic responses. A relatively reflexive process, low road transfer figures most often in near transfer (Perkins & Salomon, 1992:8).
Mathematical Pathology	Mathematical examples designed to violate properties that are perceived as valid, depending on the degree of sophistication of the learner. Objects “cooked up” to provide interesting examples of counterintuitive behaviour (Wolfram MathWorld, 2020).
Pathology	An experience of suffering (Sriraman & Dickman, 2017:1).
Pedagogical content knowledge (PCK)	Shulman (1986:9) defines pedagogical content knowledge as the amount and organisation of knowledge <i>per se</i> in the mind of the teacher. It is the teacher’s interpretations and transformations of subject-matter knowledge in the context of facilitating student learning (Van Driel Verloop & De Vos, 1998:673).
Piaget learning	“Learning without being taught”. The concept of cognitive structure is central to his theory (Papert, 1980:7).
Promote	The Merriam-Webster dictionary defines promote as “to contribute to the growth or prosperity of” (Merriam-Webster.com, 2020).
Object-oriented programming	Object-oriented programming (OOP) is a software programming model constructed around objects. This model compartmentalises data into objects (data fields) and describes object contents and behavior through the declaration of classes (methods) (Techopedia.com, 2017).
Scratch	Scratch has a simpler object model without classes which allows a very quick and easy start, but has drawbacks for some kinds of more advanced projects, especially those involving many objects of the same kind (Kölling, 2010b:20).

CHAPTER 1: INTRODUCTION

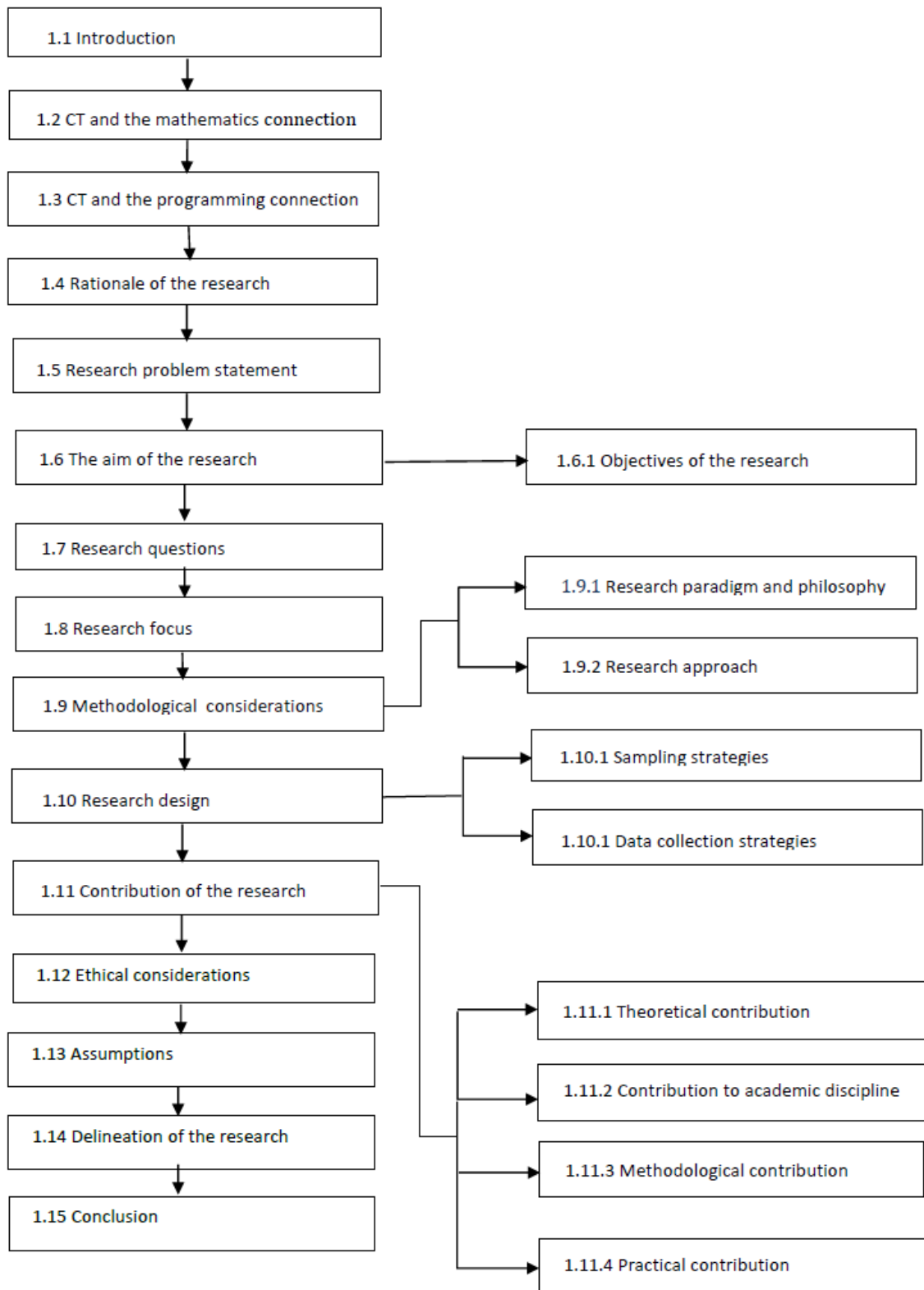


Figure 1.1: Chapter 1 layout

1.1 Introduction

The underperformance of learners¹ in especially mathematics at high schools in South Africa (SA) remains a challenge for government, policymakers and educators (Reddy et al., 2015; Voogt et al., 2015; Reddy, 2014; Spaul, 2013). The Department of Basic Education (DBE) in SA has been embarking on a National Development Plan (NDP) with a five-year strategic plan from 2015 to 2020, to enhance mathematics education among learners. In SA, systemic tests have been introduced since 2012 as an upgrade of the Annual National Assessments (ANAs) instruments (Schäfer, 2018), but computational thinking is not specifically addressed. The implementation, in general across the world, to enhance the status quo of mathematics education, remains complex, as highlighted by Voogt et al. (2015), Denning (2017), and Lockwood and Mooney (2017).

Subjects such as Mathematics and Science expect from learners to operate effectively at a cognitive level of formal operations in performing computational thinking. Computational thinking demands thought processes, abstraction and decomposition (Guzdial, 2008; Wing, 2011; Selby & Woollard, 2014). Computational thinking is initially defined as that what “involves solving problems, designing systems and understanding human behaviour, by drawing on the concepts fundamental to computer science” (Wing, 2006:33).

Teaching towards mathematical problem solving remains a challenge, according to Chirinda and Barmby (2018). Chirinda and Barmby (2018:122) consider mathematics to be “inexplicable and beyond understanding for the majority of learners”. These challenges exist due to many factors within the South African society, such as large or overcrowded classrooms and beliefs about mathematics in general (Moscucci & Bibbo, 2015). Lockwood and Mooney (2017) opine that detailed lesson plans and curriculum structures are lacking in terms of computational thinking.

Teachers using a computational thinking educational framework seldom assess competencies, but more often the “way of doing things” (Denning, 2017:36). The author argues that the learner must be involved in the design of computations to become a computational thinker as opposed to using instruments that do it for the learner. Similar arguments are held by Lee et al. (2011), where the processes to create automations are seen as computational thinking and not the interactions with automations as in the case of robotics rollouts in education. Lockwood and Mooney (2017:1) regard the interfacing of

¹ The statistics showed the decline in the overall grades of Grade 9 learners (DBE, 2015; HSRC, 2014; Spaul, 2013). According to Spaul (2013) and Reddy et al. (2015), the Literacy and Numeracy Assessment (LaNA) rolled out by the Trends in International Mathematics and Science Study (TIMSS) indicates that South African learners are underperforming and hold the last position out of a total of 143 countries (Reddy, 2014).

computational thinking and specific subjects as a work in progress and very much in its “infant stage”. The authors reference multiple examples from all disciplines, but identified a programming language approach to integrate computational thinking into the specific discipline or in using some Computer Science (CS) principles within the discipline. This approach is contradictory to that of Voogt et al. (2015) who do not view programming as the sole source of computational thinking and above all, learners may not be comfortable when programming is used. Denning (2009, 2017) has perceived computational thinking as a hallmark of CS since the 1950s. Denning (2017) further posits that he has doubts about statements made since 2006 to promote computational thinking to all K-12 educational institutions, also known as “computational thinking for all” (Wing, 2006; Barr & Stephenson, 2011; Deschryver & Yadav, 2015, Yadav et al., 2016; Heintz & Manilla, 2018). Denning argues that other domains offer vague and confusing definitions of computational thinking, and acknowledges the challenge in the definition and assessment of computational thinking within these domains. Researchers such as Heintz and Manilla (2018) perceive “computational thinking for all” in context of using programming being rolled out to all, instead of applying computational thinking in subjects other than CS.

1.2 Computational thinking and the mathematics connection

When looking at the mathematical problem-solving approach within SA, Chirinda and Barmby (2018) find that the DBE’s adoption of such a rollout is not properly understood by learners and teachers. Selby and Woollard (2014) opine that mathematical thinking points to abstract structures and computational thinking to an abstract methodology. One of the reasons for the misunderstanding between the DBE, teachers and learners in SA, is that computational thinking is lacking among the majority of learners as indicated by low pass rates in mathematics. Reflective abstraction and algorithms form the basis of the mathematical problem-solving approach (Aho, 2012; Cetin & Dubinsky, 2017; Denning, 2017; Chirinda & Barmby, 2018). According to Selby and Woollard (2014), computational thinking points to an abstract methodology and mathematical thinking to abstract structures. The Action Process Object Schema (APOS) theory is well researched on how mathematics learning takes place (Arnon et al., 2014) and has a strong connection with computational thinking embedded in reflective abstraction (Cetin & Dubinsky, 2017).

1.3 Computational thinking and the programming connection

The initial definition of computational thinking by Wing (2006:33) states that it “involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science”. Voogt et al. (2015) argue that computational thinking in CS requires knowledge of programming and that computational thinking concepts are developed through CS. This viewpoint ties in with Wing’s computational thinking

automation and abstraction definition (Wing, 2011). According to Aho (2012) and Denning (2017), abstraction refers to thought processes that formulate a solution. Automation executes the solution using a computational model (Aho, 2012; Denning, 2017). The debate revolves around the computational model being only a machine or any entity. The teaching of computational thinking is governed by the methodology of instructors on the higher-level concepts in multiple domains. “Teachers must be shown how computational thinking can enhance their teaching and their students’ understanding of their content area” (Kale et al., 2018:575). These authors further argue that successful application of technology to specific teaching of content is still a challenge.

As stated earlier, Voogt et al. (2015) posit that the application of computational thinking in multiple domains should deviate from Wing’s (2006) definition of computational thinking, namely that learners should not think as computer scientists do, but stay within their own discipline. This makes delineation of computational thinking within multiple domains a challenge when compiling a curriculum. Denning (2017) points out that computational thinking is not only as Wing (2006) intended it to be, where education embraces the term ‘computational thinking’ in general, but views computational thinking as a process or activity whereby researchers or implementers of computational thinking are engaged in using computational thinking when a solution to a problem is sought to establish a computational model (Aho, 2012; [Figure 2.10](#)). The educational model for teaching and learning in SA should be adjusted to focus on the development of competencies by eliminating all the academic noise that sometimes clutters competency development.

This chapter further focuses on the (i) rationale of the research, (ii) problem statement, (iii) research questions, (iv) aim of the research, (v) research methodology, (vi) contribution of the research, (vii) ethical approach, (viii) assumptions, and (ix) delineation of the research, which will be discussed in the next sections.

1.4 Rationale of the research

A magnitude of challenges exists at all levels of the pedagogical space when attempting to provide an ideal learning environment for all learners. Some of these challenges include the (i) different approaches to teaching and learning at schools, (ii) learner beliefs about mathematics, and (iii) availability of resources. Other challenges are (iv) discipline at schools and communities, (v) lack of respect for others, and (vi) basic needs such as electricity and sanitation. Reddy et al. (2015) posit that poor discipline, violence and bullying at schools have a negative effect on the performance of learners at a cognitive level of formal operations.

Poor and middle class (economical) learners in grade 8 are seen as a barometer to predict the pass rate in Grade 12. Raising the mathematical scores in Grade 12 implies raising the scores in grade 8, especially in the case of poor academic performance. To achieve this raising of mathematical scores, attention should be given to mathematical problem solving from grade 8 onwards (Chirinda & Barmby, 2018). The educational playfield, when looking for example at a subject such as Mathematics that requires computational thinking, is at risk and the development of computational thinking requires urgent attention among high school learners. Chirinda and Barmby (2018) highlight that an algorithmic approach is foreign to the majority of learners and teachers, and a clear systematic approach in promoting computational thinking is needed to deal with algorithms and reflective abstraction (Cetin & Dubinsky, 2017).

1.5 Research problem statement

Spaull (2013), Reddy (2014) and Reddy et al. (2015) argue that there are not enough learners in subjects such as Mathematics and Science where computational thinking is required to satisfy the demand and expectations of industry and society. The manner in which learners and students engage with their studies does not show sufficient support for the development of computational thinking at a cognitive level of formal operations (Voogt et al., 2015; Denning, 2017; Lockwood & Mooney, 2017). This phenomenon is mainly due to the underperformance of learners. Computational thinking needs to be understood and learners should engage with computational thinking in such a way that their computational thinking skills are promoted at a cognitive level of formal operations, but it remains a challenge to achieve (Voogt et al., 2015; Denning, 2017; Lockwood & Mooney, 2017). Unfortunately, little research has been done on how computational thinking is promoted among high school learners at a cognitive level of formal operations.

The problem statement is formulated as follows: It is unclear how computational thinking can be promoted among high school learners at a cognitive level of formal operations.

1.6 Aim of the research

The aim of this research was to explore and understand how a programming language, using Action Process Object Schema (APOS) theory as lens, could promote computational thinking skills at a cognitive level of formal operations among high school learners.

1.6.1 Objectives of the research

In order to realise the aim of this research, the objectives of the research are the following:

- i) To determine what computational thinking concepts are, and the role they play at a cognitive level of formal operations.

- ii) To determine the characteristics of a typical programming language that may promote computational thinking skills at the cognitive level of formal operations.
- iii) To determine the factors which promote computational thinking among high school learners at a cognitive level of formal operations (CLFO).
- iv) To determine higher-level constructs within a programming language which promote APOS among high school learners.
- v) To combine the usage of an LMS and a programming language in order to assist high school learners with “worked examples” of advanced higher-level constructs in a programming language and cognitive load theory.

1.7 Research questions (RQs)

To achieve the objectives, the researcher formulated two research questions (RQs), together with sub-research questions (SRQs), to generate answers for the problem statement, namely:

RQ 1: What are the characteristics of an enhanced learner’s teaching and learning strategy that can empower learners to master computational thinking skills through APOS theory, infused by a programming language at high school level?

RQ 2: How can computational thinking skills at a cognitive level of formal operations be promoted among high school learners through the teaching of a programming language aligned to Action Process Object Schema (APOS)?

The RQs, SRQs, methods to be used to answer the RQs as well as the objectives of each question are presented in [Table 1.1](#) and [Table 1.2](#).

Table 1.1: Mapping of research methods and objectives to RQ 1

RQ 1: What are the characteristics of an enhanced learner’s teaching and learning strategy that can empower learners to master computational thinking skills through APOS theory, infused by a programming language at high school level?		
SRQs	Research Method	Objective
1.1: What factors are needed to promote computational thinking at a cognitive level of formal operations among high school learners?	Literature Analysis	To determine the factors which promote computational thinking among high school learners at a cognitive level of formal operations (CLFO).
1.2: What type of programming language may be used to promote computational thinking skills at a cognitive level of formal operations?	Literature Analysis	To determine the characteristics of a typical programming language that may promote computational thinking skills at the cognitive level of formal operations.
1.3: What constructs within the programming language facilitate APOS theory at a cognitive level of formal operations?	Literature Analysis	To determine commonalities of constructs in APOS and a programming language.

Table 1.2: Mapping of research methods and objectives to RQ 2

RQ 2: How can computational thinking skills at a cognitive level of formal operations be promoted among high school learners through the teaching of a programming language aligned to Action Process Object Schema (APOS)?		
SRQs	Research Method	Objective
2.1: How are the constructs of a programming language taught among high school learners at a cognitive level of formal operations?	Education DR	To explore and understand how constructs of a programming language facilitate high school learners at a CLFO
2.2: How do the constructs of a programming language align to APOS among high school learners at a cognitive level of formal operations?	Education DR	To determine higher-level constructs within a programming language which promote APOS among high school learners
2.3: How does the use of an LMS, as a platform for learning, aid the teaching of a programming language aligned to APOS to promote computational thinking skills at a cognitive level of formal operations among high school learners?	Education DR	To combine the usage of an LMS and a programming language in order to assist high school learners with “worked examples” of advanced higher-level constructs in a programming language and cognitive load theory (CLT)

1.8 Research focus

This research focuses on the ages of learners in grades 8 and 9, which forms part of the “Abstract-Coordination-Hierarchisation” stage within the Neo-Piagetian model of development as depicted in Figure 1.2 (Young, 2012:244). The abstract and collective intelligence stages are analogous to that level of formal operations, which is a broader classification as inspired by Piaget’s (1977:121) original research.

A model of 25 steps in Neo-Piagetian cognitive development and Neo-Eriksonian social-affective development.

Level	Neo-Piagetian stage	Substage	Age range	Neo-Eriksonian stage	Neo-Eriksonian substage
1	Reflexive	Coordination	Earlier fetal life	Non-participatory reflexive socio-emotions	Distance acts vs. no acts
2		Hierarchization	Quite premature		Nursing vs. rootless acts
3		Systematization	Somewhat premature		Outcome vs. outcast acts
4		Multiplication	Full-term newborn		Caregiving vs. careless giving acts
5		Integration	0-1 Month		Emotional vs. malemotional acts
6	Sensorimotor	Coordination	1-4 Months	Pre-participatory socio-affects	Dyadic vs. dysdyadic acts
7		Hierarchization	4-8 Months		Trust vs. mistrust acts
8		Systematization	8-12 Months		Sociability vs. unsociability acts
9		Multiplication	12-18 Months		Autonomy vs. doubt acts
10		Integration	18-24 Months		Interdigital vs. dedigital acts
11	Perioperational	Coordination	2-3.5 Years	Peri-participatory social cognitions	Superordinate vs. discoordinate acts (quasi-participatory)
12		Hierarchization	3.5-5 Years		Initiative vs. guilt acts
13		Systematization	5-7 Years		Identification vs. problematic identification acts
14		Multiplication	7-9 Years		Industry vs. inferiority acts (participatory)
15		Integration	9-11 Years		Role vs. role confusion acts
16	Abstract	Coordination	11-13 Years	Hyper-participatory social mutuality	Conscious vs. contraconscious acts
17		Hierarchization	13-16 Years		Identity vs. identity diffusion acts
18		Systematization	16-19 Years		Nurturing vs. misnurturing acts
19		Multiplication	19-22 Years		Intimacy vs. isolation acts
20		Integration	22-25 Years		Universal vs. self-singular acts
21	Collective intelligence	Coordination	25-28 Years	Superordinate participatory collective sociality	Metacollecting vs. disillusionment acts
22		Hierarchization	28-39 Years		Generativity vs. self-absorption acts
23		Systematization	39-50 Years		Catalytic vs. midlife crisis acts
24		Multiplication	50-61 Years		Ego integrity vs. despair acts
25		Integration	61 Years		Cathartic vs. abandonment acts

Figure 1.2: A 25-step model in Neo-Piagetian cognitive development and Neo-Eriksonian social-affective development (Adopted from Young, 2012:242)

Ojose (2008) identifies a challenge for learners at a cognitive level of formal operations (CLFO) to connect mathematics with activities they do. This CLFO requires reasoning where “reasoning skills within this stage refer to the mental process involved in the generalising and evaluating of logical arguments and include clarification, inference, evaluation and application” (Ojose, 2008:28).

1.9 Methodological considerations

A brief overview of the following methodological considerations governing the research will be discussed next. These considerations are: (i) research paradigm and research philosophy; (ii) research approach; (iii) research design; (iv) sampling strategies; and (v) data collection strategies.

1.9.1 Research paradigm and research philosophy

Freshwater and Cahill (2012) as well as Saunders, Lewis and Thornhill (2019) argue that research philosophies are investigated through research paradigms. The overall research philosophy points to the development of knowledge and the nature of that knowledge (Saunders, Lewis & Thornhill, 2019). Burrell and Morgan (1979) and Guba and Lincoln (1994) view a paradigm as a set of basic beliefs. When criticising the term “belief”, D’Amore’s (2008:3) definition states that a belief is “an opinion, set of judgements and of expectations, which one thinks with regards to something”.

According to Burrell and Morgan (1979), certain assumptions are made by the researcher, which revolve around the realities within the research (ontology), human knowledge (epistemology) and how the values and beliefs influence the research process (axiology). The ontology for Design Science Research (DSR), according to livari (2007), should be based on Popper’s three worlds, of which World 3 is the significant world based on human artefacts and institutions and theories within the ambit of this research. Cole et al. (2005) perceive Design Science (DS) and Design Research (DR) as synonyms in the context of Information Systems (IS) research. According to Venable and Baskerville (2012:142), DSR points to a purposeful artefact for humans which can be “a product or a process; it can be a technology, a tool, a methodology, a technique, a procedure, a combination of any of these, or any other means for achieving some purpose”. This research addressed a significant educational challenge and hence the DR method is that of Educational Design Research (EDR) (Nieveen, 2013; Plomp, 2013; Bannan, 2013; Van Wyk & De Villiers, 2018). Saunders, Lewis and Thornhill (2019) represent the traditional research process with the research onion as depicted in [Figure 1.3](#).

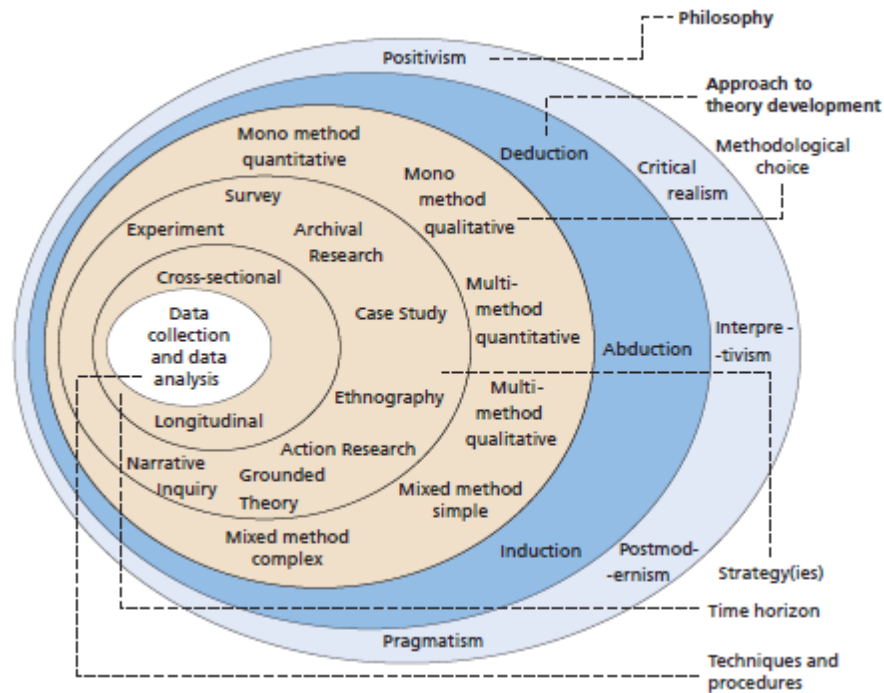


Figure 1.3: Research Onion (Adopted from Saunders, Lewis & Thornhill, 2019:130)

According to Saunders, Lewis and Thornhill (2019), a research philosophy is a reflexive process because of too many variables and the subjectivity of the researcher, which influences the research philosophy of the researcher. When criticising the research onion, relative to this research, this research uses an interpretive research philosophy by adding to the theory through an inductive approach using a research strategy namely EDR. Saunders, Lewis and Thornhill (2019) did not include DSR and EDR as a research strategy. The reasons for this are not clear, except that their research philosophy focuses on business as opposed to education.

1.9.2 Research approach

According to Saunders, Lewis and Thornhill (2019), there are several approaches to conduct research, namely deduction, induction and abduction. Creswell (2014), on the other hand, recognises quantitative, qualitative and mixed methods as three approaches. This research strategy was that of EDR and built on kernel theories that described the Information Technology (IT) artefact (Weber, 2010). These kernel theories were established on “the evaluation and modification of the natural and social theories” (Weber, 2010:2).

Within EDR, the worldview is regarded as a mixed paradigm rather than a single paradigm. According to Vaishnavi and Kuechler (2008), Weber (2010), Beck and Weber (2013), and Gregor and Hevner (2013), the instantiation of the artefact can be a construct, model, method, instantiation or combinations of the former. An artefact is made with a purpose; it can be a construct, model, framework, solution instantiation, or theory (Vaishnavi & Kuechler

2008; Beck & Weber, 2013; Gregor & Hevner 2013). Weber (2010) further argues that the socio-technologist or developmentalist paradigm forms part of the EDR paradigm. EDR in its full design life cycle consists of a three-part research paradigm. This research is positioned as part of an interpretivist/constructivist paradigm to create an IT artefact as discussed in the next section.

1.10 Research design

Research design refers to the way the research problem is investigated (Maree, 2012). This research explores and understands how a programming language, using the Action Process Object Schema (APOS) theory as lens, may promote computational thinking skills at a cognitive level of formal operations among high school learners. Learners of a private high school were the subjects who took part in the research, and the sample size of 15 to 20 learners per class was informed by the research design. The only difference between the learners was the separation based on language. Furthermore, learners followed the same curriculum in grades 8 and 9. At that stage of their schooling, they took the same subjects overall and formed a homogenous group.

1.10.1 Sampling strategies

A non-probability, purposive sampling method was used to select the sample of research participants within this qualitative research (Maree, 2012). Sharma (2017) states that purposive sampling relies on the judgement of the researcher and is subjective for it is prone to researcher bias. Although new-Piagetian perspectives are noted (Suizzo, 2000; Rutherford, 2011; Young, 2012; Barrouillet, 2015), Piaget's (1977) stage development model was chosen. Learners fall within the cognitive level of formal operations (CLFO), which refers to individuals of ages 12 years to adulthood. This correlates with the average age of grade 8 and 9 learners. The researcher accepted the Piagetian model of choice based on the research sample (Piaget, 1964; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki, & Khoshhal, 2017). The sampling method aligned with the theoretical conceptual framework of this research (section 2.2.3, Figure 2.21). The sample of learners, which constituted classes of English and Afrikaans speaking learners, separated only by language, was chosen by the IT teacher of the school. This research replaced IT classes. The IT teacher selected the participating IT classes based on availability of time slots to conform to the practical day-to-day operation of the school without disrupting classes. To mitigate researcher bias, as mentioned by Sharma (2017), the IT teacher did not make any groupings based on performance; two classes, one for Afrikaans speaking learners and one for English speaking learners, were selected as the unit of analysis.

1.10.2 Data collection strategies

The interaction with the same groups of learners spanned across two years, and lessons were prepared based on the genetic decomposition of Dubinsky and McDonald (2001) (Arnon et al., 2014). The research strategy was EDR using a programming language with APOS theory as lens. Greene, Caracelli and Graham (1989), Onwuegbuzie and Collins (2007), Venkatesh, Brown and Bala (2013) and Creswell (2014) suggested two frameworks to collect data either concurrently or sequentially. This research used a sequential design, focused on the promotion of computational thinking among high school learners. Data were collected through lectures, semi-structured interviews, observation, assessments and tasks received from learners. The interviews and outcomes of the assessments and assignments were transcribed and provided insight into learner perceptions on certain concepts such as abstraction. Interviews were done in a semi-structured manner based on a semi-structured questionnaire (Appendices R1, R2 & R3). Learners interacted with one another and they were observed when using a personal computer (PC) and the Greenfoot programming language. These interactions were also captured on video and then analysed. The coding produced by the learners can be seen as documentation that showed their competency skills, highlighting strengths and weaknesses within the programming activities, supported by the analysis of assessments of these programming activities.

1.11 Contribution of the research

Te'eni et al. (2015:564) state three success components for research, namely “contribution, contribution and contribution”. The authors are critical of theoretical and empirical contributions and urge researchers to make findings exciting by contextualising it outside the scientific study as well. This research study contributed to the existing body of knowledge through four types of contributions, namely, (i) theoretical, (ii) academic discipline knowledge, (iii) methodological, and (iv) practical contributions, which will be explained in the following sub-sections (Hofstee, 2009; Jansen, 2012).

1.11.1 Theoretical contribution

The researcher investigated the use and application of programming concepts, through APOS theory as lens, to promote computational thinking skills among high school learners at a cognitive level of formal operations (section 8.2.1). Teachers and researchers may promote computational thinking by following the protocol as developed in this research.

1.11.2 Contribution to academic discipline

According to Maree (2012), academic disciplines grow when participants and the participating context are enriched through research (section 8.2.2). The learners and

researcher gained qualities through research, such as developing new skills that enrich their personal environment, the school environment and the broader community.

1.11.3 Methodological contribution

Many methodologies were researched, compared and applied to develop tools for this research. Methodologies such as Schema development and research methodologies; technical, pedagogical and content frameworks to develop curricula; and meta-cognitive methodologies to address belief systems were combined to develop a framework. A proposed conceptual framework (section 8.4.2, Figure 8.4) was constructed from the initial theoretical conceptual framework (section 2.2.3, Figure 2.21) to facilitate the usage of Greenfoot as a programming language in a controlled constructionist manner (section 2.2.2.1(b)).

1.11.4 Practical contribution

This research produced a guideline towards a radical change and the regulation dimension in provisioning the rolling out of a new curriculum parallel with the existing curriculum on the Greenfoot programming language and APOS theory. The abstract part (radical change), according to Cronje (2016), which explores programming language concepts to determine which programming language concepts were of importance to assist with computational thinking, were addressed (section 8.2.4). Target audiences such as teachers and researchers may follow protocol to replicate this research and promote computational thinking among learners.

1.12 Ethical considerations

The Research and Ethics Committee of the Cape Peninsula University of Technology within the Faculty of Informatics and Design granted permission to conduct this research (Appendix A-2). Leedy and Ormrod (2014) state that ethical issues can be classified into the following four categories: (i) protection from harm; (ii) informed consent; (iii) right to privacy; and (iv) honesty with professional colleagues. The researcher ensured that informed consent was obtained from the schools as a starting point (Appendix A), and that the learners and parents were informed of daily activities the researcher performed with the learners. Privacy of the research participants was ensured at all times (Leedy & Ormrod, 2014). Consent was also obtained from Oracle Academy in South Africa prior to commencement of the study. The names of the participating learners were encoded to ensure confidentiality between teacher-parent-learner. The school was given the option to reveal its name in this thesis. The private school was consulted to obtain permission before conducting the proposed study (Appendix A-1). In a follow up visit, the teachers and the principal were consulted to clarify the aim of the study. The teachers were given the surety that their involvement would be kept to a

minimum so as to not add to their workload, but the incentive of having an enhanced educated learner was a convincing strategy which tied in with the contribution of academic discipline. General data, interviews, videos and questionnaires are stored on a separate hard drive protected by virus (Norton) and anti-malware software (Malwarebytes).

1.13 Assumptions

It was assumed that the age group 11 to 15 years is a relevant unit of analysis for the research according to Piaget's (1977) cognitive level of formal operations classification states the age of 12 to adulthood (Piaget, 1964; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017). It was further assumed that the grade 8 and 9 learners were of the same emotional developmental stage. Owing to the researcher's subjectivist ontological stance and the teaching of programming language concepts in Greenfoot, it was assumed that the researcher would play a part in the outcome of the research. In the case of the chosen private school, their approach to learning was based on a problem-solving model, and the application of Greenfoot provided a starting point in the Java programming language. Consideration that Java was used in Grade 11 and Grade 12 and the research addressed their problem-solving model, the research was accepted.

1.14 Delineation of the research

The study was conducted at a private school where the infrastructure supported the research initiative, i.e. teachers and technology. The private school was identified based on their accessibility and because Java was installed on their computers. Public and state schools were excluded from the study because the DBE abandoned Java, which affected resources and the vertical articulation of the research as put forward as an advantage to conduct the research.

1.15 Conclusion

The problem, namely it is not clear how computational thinking can be promoted among learners to facilitate subjects that demand computational thinking, has been discussed. Mathematical problem-solving deals with abstract structures, and computational thinking focuses on an abstract methodology. APOS theory is a well-researched mathematical model, framework and theory. Reflective abstraction forms part of APOS theory, which acts as scaffold for computational thinking at methodological level. Computational thinking is poorly structured in curricula and is mainly found within CS, which contradicts the "computational thinking for all" vision among educationists. The aim of this research was therefore to explore and understand the role of a programming language's concepts and abstractions through APOS theory in order to promote computational thinking. This was done by introducing programming language concepts using the APOS theory from a mathematical perspective,

underpinning the research, to grade 8 and 9 learners at a private high school in South Africa. The motivation is found in computational thinking being an abstract methodology and APOS theory being a model and framework. Grade 8 and 9 learners fall within the age group of 11 to 15 years, aligned to Piaget's (1977) development stage of the cognitive level of formal operations. Researchers highlight numerous challenges in administering computational thinking, but differ in their viewpoints. Computational thinking is vaguely described in mainly CS curricula, which contributes to more confusion in the rollout of computational thinking and assessment of competencies. Detailed lesson plans are lacking. Many researchers or educators only view computational thinking in relation with programming and CS, which creates a negative perception and complicates computational thinking rollout even more. To remedy all the negative reasoning, this research was facilitated by detailed lesson plans. Learners were involved with designing computations that strengthen computational thinking rather than using or interacting with instruments such as robotics only. Programming should not be perceived as a quick remedy to increase mathematics results, but rather as a long-term fixing strategy in changing the traditional system of beliefs about mathematics of the learner.

1.16 Summary

The acquisition of computational thinking involves learners' thought processes, dealing with abstraction and decomposition at a cognitive level of formal operations. Computational thinking is a necessary competency needed to comprehend subjects such as Mathematics and Science. The implementation of computational thinking among learners within the DBE is either vague or non-existent.

This research aimed to explore and understand how a programming language, using Action Process Object Schema (APOS) theory as lens, could promote computational thinking skills at a cognitive level of formal operations among high school learners. As objective, the researcher wanted to determine how computational thinking skills at a cognitive level of formal operations could be promoted among high school learners by teaching a programming language aligned with APOS theory.

EDR as research strategy was adopted using an induction-based approach through the lens of a conceptual theoretical framework. Data were collected through lectures, semi-structured interviews, observation, assessments and tasks received from learners. The interviews and outcomes of the assessments and assignments were transcribed and analysed to provide insight into learner perceptions on certain concepts such as abstraction. Semi-structured interviews were conducted based on a questioning plan in a conversational style. Interactions between learners were also captured on video and analysed. The coding

produced by the learners was used as documentation that portrayed their competency skills, highlighting strengths and weaknesses within the programming activities, supported by the analysis of assessments on these programming activities.

The research methodology was based on an interpretivist research philosophy. The ontological underpinning was subjective and the epistemological stance accepted opinions of learners through written, spoken and visual attributed meanings. The axiology of the researcher was that of a practicing educator in programming, a teaching and learning expert, an instructional designer for the Independent Institute of Education, and a certified Java-Alice-Greenfoot instructor through the Oracle academy.

All the research activities were based on sound ethical standards as maintained and condoned by the Cape Peninsula University of Technology at the Faculty of Informatics and Design (Appendix A-2). Informed consent was obtained from the schools as a starting point (Appendices A-1, A-3, A-4, A-5) and the learners and parents were kept in the loop.

In Chapter 2, literature on computational thinking is reviewed and analysed to cast more light on the two research questions and respective SRQs. These questions have been formulated to investigate how computational thinking may be promoted among high school learners in grade 8 and 9. The next section provides an overview on the structure of the thesis.

1.17 Structure of the thesis

The thesis consists of eight chapters as indicated in [Table 1.3](#).

Table 1.3: Structure of the thesis

Flow of the research	Logic and structure of the thesis
The research problem	Chapter 1: Background of the research, the rationale and the problem statement
The literature review	Chapter 2: Literature review
Design research	Chapter 3: Design research
Research design	Chapter 4: Research design
Data analysis and findings	Chapter 5: Data analysis and findings
Discussion	Chapter 6: Discussion
Conclusions and recommendations	Chapter 7: Conclusions and recommendations
Contributions and reflections	Chapter 8: Contribution, further research and reflections
References used in the research	References
Appendices referred to in the research	Appendices

CHAPTER 2: LITERATURE REVIEW

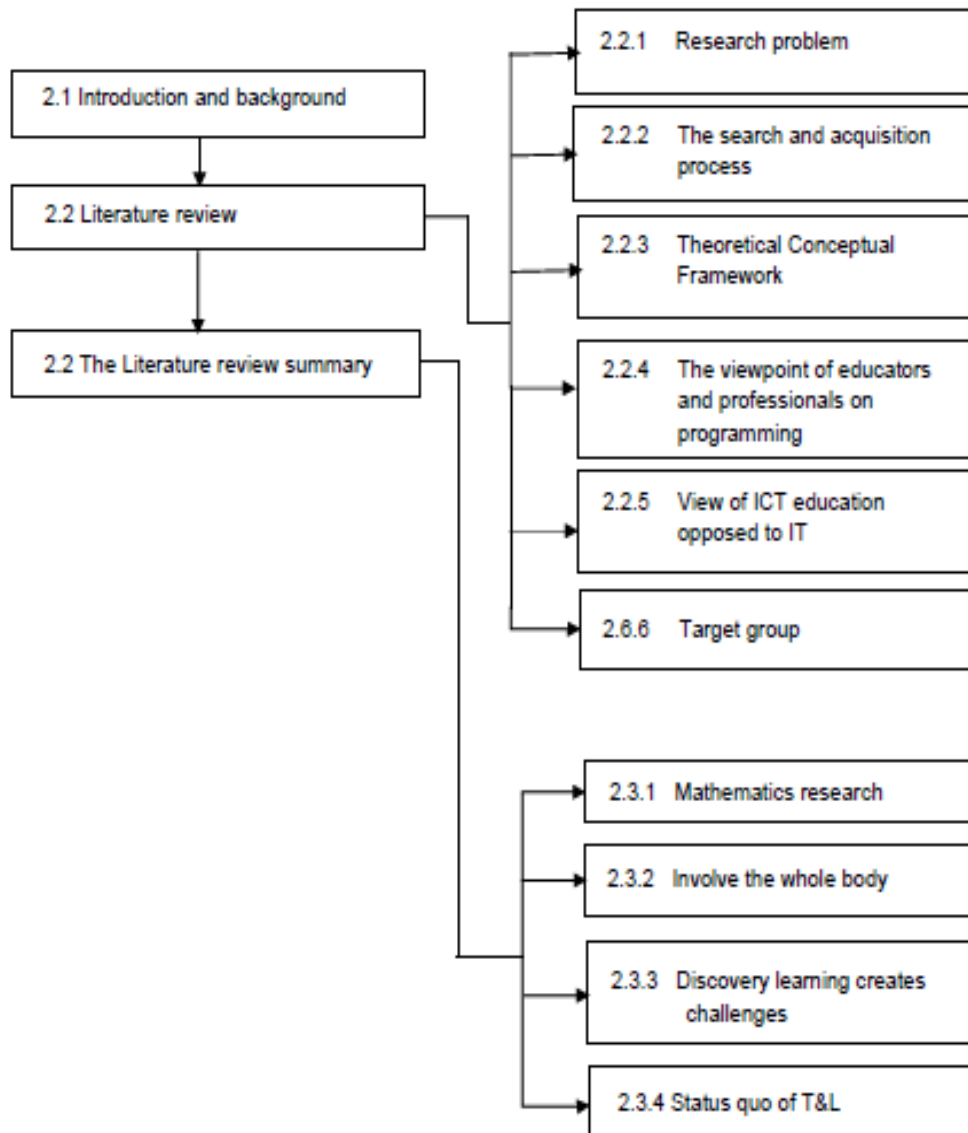


Figure 2.1: Chapter 2 layout

2.1 Introduction and background

The underperformance of learners² in especially mathematics at high schools in South Africa remains a challenge for government, policymakers and educators (Spaull, 2013; Reddy, 2014; Reddy et al., 2015; Voogt et al., 2015). Subjects such as Mathematics and Science expect from learners to operate effectively at a cognitive level of formal operations. The

² The statistics showed the decline in the overall grades of Grade 9 learners (Spaull, 2013; HSRC, 2014; DBE, 2015). According to Spaull (2013) and Reddy et al. (2015), the Literacy and Numeracy Assessment (LaNA) rolled out by the Trends in International Mathematics and Science Study (TIMSS) indicates that South African learners are underperforming and hold the last position out of a total of 143 countries (Reddy, 2014).

cognitive level of formal operations is a Piagetian stage of development from the ages of 12 to 16 (Piaget, 1964; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015).

The literature review is done by determining key words and concepts from the thesis title, problem statement, research questions and sub-research questions as well as the aim of the study. The literature chapter (Figure 2.1) is presented as follows: (i) research focus; (ii) status quo of mathematics in South Africa (SA); (iii) computational thinking and the mathematics connection; (iv) computational thinking and the programming connection; (v) the literature review; (vi) analysis and interpretation of the research questions; and (vii) summary.

2.2 Literature review

The following questions, from a mathematical perspective, where the researcher identified challenges in the South African educational system, inform why/how computational thinking skills should be promoted among learners:

- i) What framework exists to promote mathematical thinking? In answering this question, the APOS framework may cast light on understanding the problem of why there is a lack of computational thinking skills among these learners, with the focus on Mathematics.
- ii) What factors are prominent within mathematical thinking among learners? In answering this question, key concepts will be highlighted. These concepts include abstraction, APOS theory, computational thinking, constructivist and constructionist approaches, and different types of epistemologies that may enlighten the relationship between mathematical thinking and computational thinking.
- iii) How can learners be supported when involved in subjects that require computational thinking? In answering this question, major issues must be identified, such as, where is computational thinking found in our educational environment? Can a programming language together with APOS theory be used to promote computational thinking among learners?

The outcome of a search and acquisition cycle on the literature that addresses the problem of a lack of computational thinking skills among learners aligns with the problem statement. Keywords and concepts were identified from the problem statement. This led to using the research questions and aim of the study as a guideline to search the literature and conduct a literature analysis within this chapter. As strategy to interrogate the literature, a literature review, and analysis of the literature was carried out. Although there are many literature analysis frameworks available, the framework of Boell and Cecez-Kecmanovic (2014) was selected for this study. The authors recognise that there is no single specific approach to

structure a literature review, but their framework as depicted in [Figure 2.2](#) provides a definite structure for analysis.

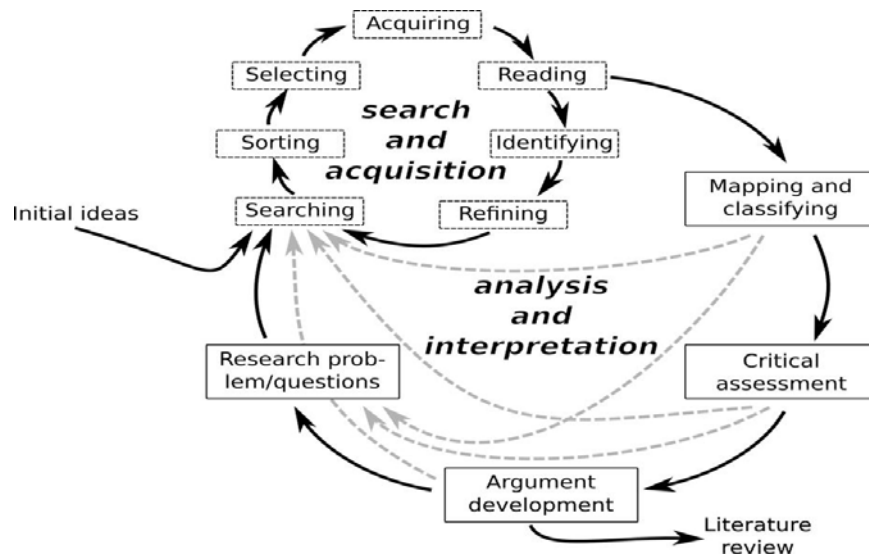


Figure 2.2: Hermeneutic framework for the literature review process (Adopted from Boell & Cecez-Kecmanovic, 2014:264)

The hermeneutic framework consists of two hermeneutic circles, namely the search and acquisition, and the analysis and interpretation. Within the search and acquisition circle, the emphasis is on the high failure rate in mathematics in Western Cape Schools as well as in the DBE at the time of the research. Upon closer inspection, Because of the low pass rates, it is not clear how the school curriculum in general contributes towards the learners' computational thinking skills to ensure a mathematical friendly mind. The term 'mathematical friendly' can be substituted with the words 'computational thinker'. The schooling system fails to deliver enough learners who are successful in any subject that requires thought processing at the cognitive level of formal operations, such as Mathematics and Science (Howie, 2004, 2013; Plomp, 2013; Maree et al., 2006; Reddy et al., 2012; Spaul, 2013; CDE, 2014; DBE, 2015; Reddy et al., 2015).

There is a shortage of learners because learners do not possess competencies in computational thinking at the cognitive level of formal operations. The low results, attributed to poor performance at tertiary institutions during the Grade 12 exit level, are confirmed by the Western Cape Education Department (WCED) (HSRC, 2014; DBE, 2015). All these factors gave rise to the research problem as discussed in the next section.

2.2.1 Research problem

The way learners and students engage in their studies does not show sufficient support for the development of computational thinking at the cognitive level of formal operations to yield enough learners who satisfy the demand and expectations of industry and society in subjects

like Mathematics and Science. Mathematics and Science curricula may be integrated owing to similar learning processes. This is supported by the National Council of Teachers of Mathematics (NCTM), the School Science and Mathematics Association, American Association for the Advancement of Science, and the National Research Council (NRC) (Bosse et al., 2010). The argument of integration revolves around the phenomena that “science provides mathematics with interesting problems to investigate, and mathematics provides science with powerful tools to use in analysing them” (Rutherford & Ahlgren, 1990: 16). The findings of Bosse et al. (2010) indicate that science subject learners may enjoy similar challenges as mathematics learners.

The problem statement is formulated as follows: It is unclear how computational thinking can be promoted among high school learners at a cognitive level of formal operations. The research questions as stated in section 1.7, Table 1.1 have been derived from this problem statement.

2.2.2 Search and acquisition process

The literature analysis commences with interrogating the problem statement and research questions through searching, reading and refining articles. These articles were critically assessed, which gave rise to a literature analysis, which, in turn, highlighted the research problem and sparked the formulation of more research questions to address the problem. Data were collected by focusing on specific researchers who were found to be spearheading their domain on these eligible articles. Key concepts, which emerged, are of an educational concern. The mathematical challenge was investigated from an educational perspective, in combination with the transfer of skills within the chosen programming language in order to promote and enhance computational thinking.

Lastly, data were analysed by providing a narrative synthesis of the data through summarising the findings from various studies and provide answers on the literature review questions. A Scopus search (Figure 2.3: Scopus Database search outcome on “Abstraction in Mathematics”) was done using the following criteria:

- i) “APOS theory” produced 392 document results with a spike between 2017 and 2018. “Computer Science” made up 5,7% and “Mathematics” 11,8% of the documents by subject area. The United States took the lead with researchers such as Trigueros, Martinez-Planell and Tayyari, while the founder, Dubinsky, creating most documents on the topic.
- ii) “Computational thinking in programming languages” yielded 310 documents with a spike in interest between 2015 and 2018. The Computer Science subject area yielded

47% of the total documents produced, which came to 250 documents, while mathematics only contributed 7%, with a total of 39 documents.

- iii) “Computational thinking” on its own yielded 3926 documents, with a spike in 2018, comprising 575 documents. The Computer Science subject area yielded 2342 documents.
- iv) “Cognitive level of formal operations” yielded 117 documents with a spike in 2012.
- v) “Greenfoot” programming language yielded 52 documents with a spike between 2009 and 2010 as well as in 2016. The well-known author, Prof M. Kölling, wrote 14 of the 26 documents.
- vi) “Abstraction in Mathematics” yielded 1231 documents of which only 496 documents were written on Mathematics as topic.

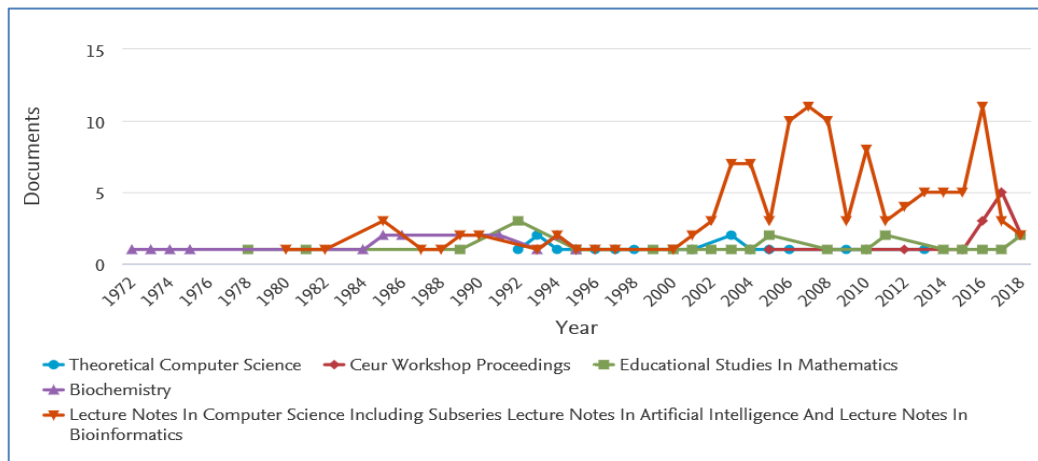


Figure 2.3: Scopus Database search outcome on “Abstraction in Mathematics”

The literature review highlights a mathematical approach in the form of APOS theory supported by the Scopus search. Further to the Scopus search, APOS theory and computational thinking combined with a programming language show a need for more research as highlighted by the problem statement. Types of abstraction were discovered that: (i) affect mathematical learning, (ii) extend abstraction towards computational thinking, and (iii) conveying the importance of understanding what is meant by abstraction and automation as two pillars in supporting computational thinking, according to Wing (2006, 2008, 2011). Finally, the *status quo* of available resources is investigated together with significant research that influences mathematical thinking in practice.

Keywords such as *computational thinking*, *abstraction*, *mathematics*, *cognitive level of formal operations* and *programming language* were observed through a Scopus search that produced 1043 results. Focusing on a specific period between 2012 and 2019, the search delivered 126 results. The search was then delineated to only include *computational thinking*, *abstraction*, and *cognitive levels of formal operations*, which yielded 502 results. Using

Google Scholar and the CPUT Library Databases, another search yielded 1200 articles. These articles included those found in the Scopus search, which were downloaded and read to determine their relevance to the topic and research questions of this study. From this, 450 articles were used for the analysis, which produced 238 references but increased to 350 as the researcher reflected on a network of contributing issues in the mix.

Using the databases available at the CPUT Library, i.e. EBSCOhost, ACM Library, Business Source Premier, Career and Technical Education, Emerald, ERIS, Google Scholar, and IEEE Xplore Digital Resources such as the Library, ProQuest, ProQuest Computing, Sabinet Reference, Science Journals, Science Direct, Scopus, and SpringerLink, to mention a few, 350 full text papers were referenced. According to Boell and Cecez-Kecmanovic (2014), this process comprises the 'Search and Acquisition' phase as part of the literature review. These articles were then analysed and interpreted as per the Hermeneutic Framework of Boell and Cecez-Kecmanovic (2014:264).

To find answers to the questions, according to Dempster and Hanna (2014), the protocol must include eligible criteria, a search strategy, valid criteria, data extraction, analysis, and dissemination properties. The eligible criteria that were searched for within articles are:

- i) Frameworks that support mathematical learning (APOS theory for this research).
- ii) The readiness level of learners, enabling them to digest activities that assist with promoting computational thinking.
- iii) Work done by researchers, which assists learners in their mathematical thinking, such as the APOS theory, beliefs about mathematics, didactic contract, didactic situation, and didactic transposition.
- iv) Classification of programming languages.
- v) Alignment of programming languages to APOS theory.

The connection between computational thinking and mathematical problem solving is highlighted by Selby and Woollard (2014), among other researchers, in section 1.2, and the promotion of computational thinking and programming (section 1.3) as an activity is embedded within the methodology. This study explored how computational thinking skills at a cognitive level of formal operations could be promoted among high school learners. The study uses a programming language and Action Process Object Schema (APOS) theory as lens to understand the exploration process.

Another reason for conducting this study was the lack of detailed lesson plans and curriculum structures (Lockwood & Mooney, 2017:1) in terms of computational thinking, and the notion that interfacing computational thinking with specific subjects is a work in progress, still in its 'infant stage'. A framework or tool to measure computational thinking and structures

to include computational thinking in the curriculum are not in place. A programming language is used when promoting computational thinking in non-CS subjects (Kale et al., 2018). The concept of “computational thinking for all” (Wing, 2006; Barr & Stephenson, 2011; Deschryver & Yadav, 2015, Yadav et al., 2016; Heintz & Manilla, 2018) to promote computational thinking to all K-12 educational institutions is challenging because of the confusion to define and assess computational thinking. That is why programming is rolled out in many studies when applying computational thinking in subjects other than CS.

2.2.2.1 Research question (RQ) 1

RQ 1: What are the characteristics of an enhanced learner’s teaching and learning strategy that can empower learners to master computational thinking skills through APOS theory, infused by a programming language at high school level?

To answer RQ 1, three SRQs were structured and analysed accordingly.

(a) Sub-research question (SRQ) 1.1

SRQ 1.1: What factors are needed to promote computational thinking at a cognitive level of formal operations among high school learners?

The research method used to determine the role of computational thinking at a cognitive level of formal operations, was a literature analysis. As indicated at the beginning of the chapter, the focus needs to be on mathematical thinking, as this is the concept that initiated the study. In answering SRQ 1.1, key factors emerged from the search and acquisition cycle, which included concepts such as abstraction, computational thinking, and constructivist and constructionist approaches to teaching and learning. Furthermore, APOS theory and different types of epistemologies that may enlighten the relationship between mathematical thinking and the role of computational thinking were investigated.

The sections in this chapter are arranged as follows: (i) Cognitive theory levels are investigated first. This is followed by a discussion on (ii) mathematical learning and embodiment, (iii) constructivism and constructionism, (iv) discovery learning challenges, (v) computational thinking, (vi) cognitive ability and computational thinking, and (vii) abstraction in action to cast more clarity as background knowledge for the relationship between abstraction and computational thinking.

(i) Cognitive theory levels

Four cognitive levels, namely, (i) the sensorimotor, (ii) pre-operational, (iii) concrete operational and (iv) formal operations level form part of the cognitive theory as created by Piaget (1975). The stage of formal operations, according to Piaget’s (1977) Cognitive

Development Theory, is the stage where the deductive way of thinking and conceptual thoughts play a role to increase intellectual capacity (Piaget, 1964; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017). It is at the level of formal operations where a person deals with abstractions, forms hypotheses, solves problems and is involved in mental manipulations (Biehler & Snowman, 1986; White & Sivitanides, 2002). The following cognitive theory levels are discussed to add clarity and position the level of formal operations in terms of Piaget's (1977) cognitive development:

◆ **Sensorimotor level (0-2 years)**

According to Piaget (1977), this stage can be subdivided into six sub-stages. These stages are: simple reflexes (0 to 1 month), primary circular reactions (1 to 4 months), secondary circular reactions (4 to 8 months), coordination of reactions (8 to 12 months), tertiary circular reactions (12 to 18 months) and early symbolic or representational thought (18 to 24 months). It comprises physical and motor practices such as basic sucking, standing and eventually mental operations that replace actions (Piaget, 1964; Young, 2012; Cherry, 2014; Ghazi et. al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017).

◆ **Pre-operational level (2-7 years)**

Biehler and Snowman (1986:62) describes this Piagetian level as “a person using his/her visual and body sensations to represent objects, but simply cannot reverse any actions”. As an example of this level, if water is poured from one container into another and the size differs, the person would judge the quantity as more or less. Imaginary play is a way by which children gain knowledge during this stage (Piaget, 1964; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017).

◆ **Concrete operational level (7-11 years)**

At this level, a person can understand conservation of matter and classifications and generalisations, but cannot understand mathematical ratios (Barker & Unger, 1983). All dogs are animals but not all animals are dogs. Children try to engage with abstract and theoretical thought (Piaget, 1964; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017).

◆ **Formal operational level (12-16 years)**

The cognitive level of formal operations is regarded as the highest cognitive level (Piaget, 1964; White, 2003; White & Sivitanides, 2002; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017). At this Piagetian level, a person must deal with abstractions, form hypotheses, solve problems systematically and engage in mental manipulations (Biehler & Snowman, 1986). Within Piaget's (1964) classification, the

cognitive level of formal operations is associated with mathematical thinking. The “Abstract-Coordination-Hierarchisation” stages within the Neo-Piagetian model of development, as depicted in [Figure 1.2](#) (Young, 2012:242), match with the formal operations stage to a degree. However, this research focused on the broader cognitive level of formal operations as stated above.

The cognitive level of formal operations is an important platform to initiate computational thinking (White & Sivitanides, 2002; White, 2003; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017). When an incorrect version of the concept, also known as a concept image (Arnon et al., 2014) is not supported by the actual definition of such mathematical concepts, a warped epistemological understanding may be reinforced. This leads to image concepts and rote learning (Dubinsky, 1991; Tall, 2004; Arnon et al., 2014). Mathematical thinking is discussed in mathematical learning and embodiment and its influence on abstraction and computational thinking in the next section.

(ii) Mathematical learning and embodiment

Tall (2004) argues that a range of theories exists in mathematics education. According to Tall (2004), researchers such as Piaget (1965), Dienes (1960) and Bruner (1966) added to the body of knowledge within mathematics. The tripartite theory of abstraction consists of three abstraction concepts, namely empirical, pseudo-empirical and reflective abstraction, which stems from Piaget’s (1964) research and is advocated by Dubinsky (1991). Dubinsky (1991) further refines his research into the APOS theory, which also forms part of the foundation of this thesis. Bruner (1966) highlights the complex world of mathematics by distinguishing three modes of mental representation, namely: the sensorimotor, the iconic and the symbolic mode, as illustrated in [Figure 2.4](#). As an individual grows, he or she goes through these phases in the form of action that is taken, then visualising those actions as thought processes, and finally, expressing him or herself through language. Visualisation is paramount towards language expression that is supported by actions as a starting point.

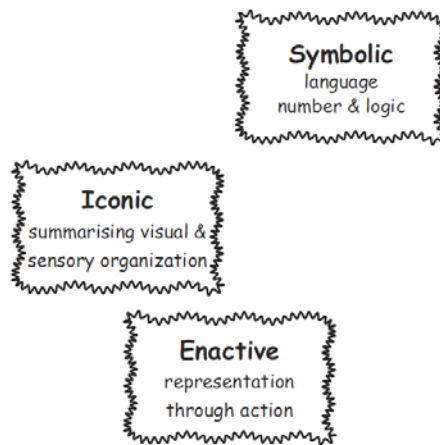


Figure 2.4: Bruner’s three modes of representation (Adopted from Tall, 2003:2)

Tall (2003, 2004) further uses the Bruner’s model and describes three mathematical worlds, namely: (i) embodied; (ii) symbolic-proceptual; (iii) and formal-axiomatic worlds. It is about our perceptions of the world, which consist of anyone’s thinking on how we perceive and sense things both physically and mentally. Psychologists such as Wilson (2002), and Wilson and Golonka (2013) use a term called ‘embodied cognition’ where the entire body and perceptually guided motions play a role in complex internal mental representations when problems are solved. This concept of including the body in learning is also highlighted through Susan Goldin-Meadow’s research conducted among deaf learners (Barrouillet, 2015) in the form of gestures. Tall (2003) incorporates the Bruner model and creates a newer model that focuses on the language aspect as well, as depicted in Figure 2.5.

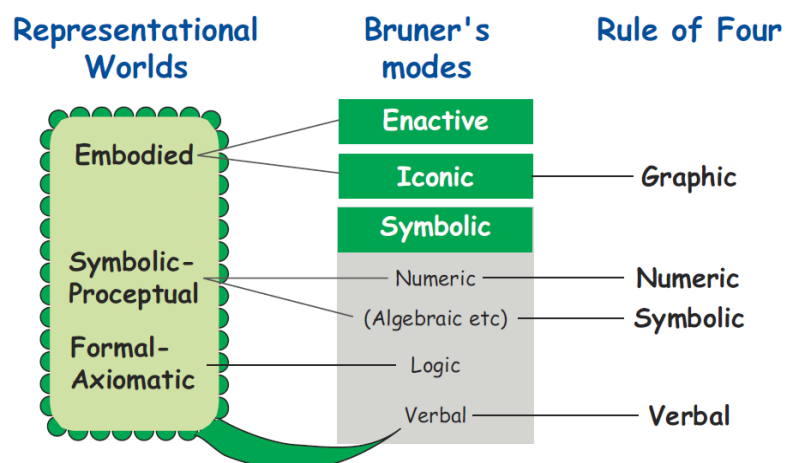


Figure 2.5: Three representational worlds and their associations with other viewpoints (Adopted from Tall, 2003:4)

Roth and Thom (2009) perceive the three worlds as the (i) conceptual embodied world, (ii) proceptual world of symbols, and (iii) the world of properties. Tall (2003, 2004) combines the inactive and visual aspects as proposed by Bruner (1966) within the embodied world.

The symbolic world consists of (i) arithmetic, (ii) algebraic and (iii) symbolic calculus, also known as the symbolic-proceptual world. A procept refers to a mental object that originates from a doable process that transforms into a thinkable concept. The world of properties described by axioms is also known as the formal-axiomatic or formal world, e.g. vector spaces are described by axioms. A sequence of theorems by inferring properties and new concepts is defined within the axiomatic system (Tall, 2003; 2004; 2008).

Figure 2.6 illustrates the human embodiment of mathematical operations of symbolism and then formalism of pure mathematics. This process then alternates back into embodiment and symbolism, but at a higher level until the learner reaches the level of formal axiomatic systems of formal proof.

The role of cognition, mathematical reasoning, learning and development of gestures were researched by Cartmill, Beilock and Goldin-Meadow (2012:131) and they refer to this as “embodied cognition”, for the body plays an important role in cognition. The authors further distinguish action from gesture, where Piaget (1977) overlooked gestures in his research (Barrouillet, 2015). Piaget (1975) brought terms such as equilibration, coordination, regulation, formation, accommodation, compensation (optimised equilibration) and assimilation to the fore (Bormanaki & Khoshhal, 2017). Movement also supports learning that ties in with gestures, and gestures may also be a type of simulated action (Cartmill, Beilock & Goldin-Meadow, 2012). The human brain is extremely limited and deals with an infinitesimal number of pieces of information at a time. The final result is a thinkable concept through compression in a category having a generic meaning to it (Tall, 2008). The Tall concept is similar to Dubinsky’s (1991) APOS theory where the end result is a schema, but in some form, the schema also becomes an object to which actions can be applied in turn (Arnon et al., 2014).

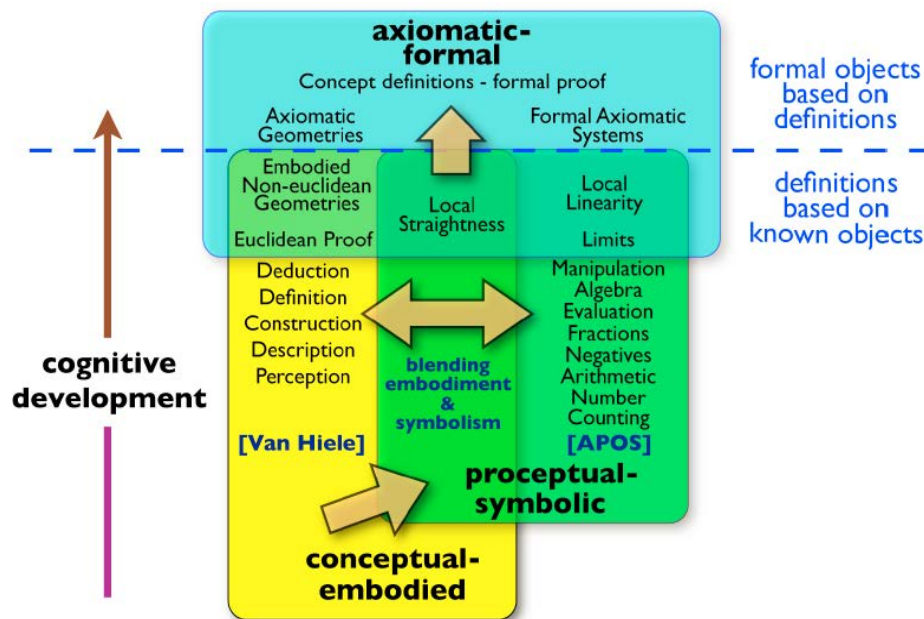


Figure 2.6: Three worlds in mathematics (Adopted from Tall, 2008:4)

Unfortunately, the Schema construction as end result is hampered by “Dyscalculia” (difficulty with numbers), which is encountered by learners and which necessitates rote learning that may become a “way of life” (Tall, 2004:286) if the learner cannot convert specific processes into thinkable concepts. Peabody (2014) views rote learning as a way to trigger learners’ investigative powers when reflection takes place, but uses rote learning only in crisis situations. “Met-befores” may cause conflicts with new contexts, and, according to Tall (2004), may necessitate rote learning. According to Bormanaki and Khoshhal (2017), Piaget (1975) used terms such as assimilation and accommodation, interpreted as the adaptation of a learner towards the concept or experience introduced to the learner at a cognitive level. The process of adaptation is done through equilibration to achieve a balance between assimilation and accommodation. Bormanaki and Khoshhal (2017) further state that equilibration is an iterative or dynamic process when experiences are assimilated and accommodated through equilibration and disequilibration. The “met-befores” highlighted by Tall (2004) may act as a false state of equilibrium to cognitively adapt when new experiences match existing schemata. Such schemata are a construct of Piaget (1975), and it can be either formal or content-based. According to Bormanaki and Khoshhal (2017:1001), a schema can be defined as “organised knowledge that one has about people, objects, places, events processes, concepts and virtually anything that provides a basis for learning”. The modality of a schema, whether formal or content, may influence the cognitive adaptation (Bormanaki & Khoshhal, 2017). Assimilation is thus strengthening a schema in terms of growth by which a learner cognitively adapts and organises the environment, which is a quantitative change.

At a workshop in March 2015, teachers were re-educated on mathematical concepts. “The training provides teachers with new methodologies, lesson plans, activities and resources for their learners” (DBE, 2015b). The DBE wants to update old information by new information, but the teachers must have schemata embedded to assimilate or grow their existing schemata, thereby creating a quantitative change. In this case, the DBE should first determine the level of these teachers’ schemata before re-educating teachers on mathematical concepts. According to Moscucci (2007), belief systems about mathematics may also influence the success of a learner’s mathematical problem solving. The DBE or any educational governing body cannot simply assume a level of education and implement refresher courses without considering mathematical beliefs among teachers and learners. Accommodation can either happen through constructing new schemata or modifying existing ones in order to match the new experience in a qualitative change. This may happen during development or growth of certain mathematical concepts among learners (Bormanaki & Khoshhal, 2017). Cognitive assimilation, cognitive accommodation, cultural assimilation and cultural accommodation are four neo-Piagetian structures put forward by Rutherford (2011). These cultural and social context issues are noted but partly ignored in this research and the focus remains on cognitive development only (by choice) when investigating teaching and learning issues.

Educators are pursuing Piaget’s (cited by Woolfolk, Winne & Perry, 2003) constructivist approaches in mathematical learning. Piaget classifies thinking into two processes: organisation and adaptation (Woolfolk et al., 2003; Bormanaki & Khoshhal, 2017). Adaptation refers to an adjustment that involves Piaget’s assimilation and accommodation processes. The process of organisation “deals with the combining, arranging, recombining, and rearranging of the behavior and thought in the coherent system” (Bormanaki & Khoshhal, 2017:997). Dis-equilibration is a state of cognitive conflict and stress, when a balance is not reached between assimilation and accommodation, known as the process of equilibration (Cook & Cook, 2005).

Constructivism and constructionism are discussed in the next section to resolve the issue of rote learning and instil an understanding or adaptation of mathematical concepts.

(iii) Constructivism and constructionism

The question regarding constructivism is, “What is constructivism and why has constructivism become hegemonic within education?” The debate and implementation of constructivism became analogous to a secular religion (McPhail, 2016:297). Piaget uses the term constructivism and Papert uses the term constructionism (Ackermann, 2001). Constructivism expresses the theory “that knowledge is built by the learner, not supplied by

the teacher”, as intended by Piagetian learning (Ackermann, 2001:4). Constructionism is “an idea that happens especially felicitously when the learner is engaged in the construction of something external or at least sharable” (Ackermann, 2001:4).

Papert (1993) further argues that emphasis needs to be placed on social constructivism and psychological constructivism. Social constructivism focuses on the nature of human knowledge around construction and survival, which points to epistemology. It is mainly curriculum content that is affected by this. Because of knowledge being a human construct, it is more relative than absolute and hence the actual existence of knowledge as the “truth” is debatable. McPhail (2016) argues that constructivism denies human knowledge as an objective picture of the world out there. It is almost a collapse of reality into the experience of the individual’s reality. Learners’ understanding of mathematical concepts is verified in their answers, but this verification is seldom correct. The learner’s understanding is more relative than absolute when compared to the correct answer.

Psychological constructivism focuses on how humans learn in constructing their own meanings and understandings (McPhail, 2016). From email communication between Dubinsky (2018) and the researcher it can be deduced that Dubinsky does not use the term ‘constructivist’ as opposed to ‘constructionist’, but supports the idea of constructionism.

(iv) Discovery learning challenges

Teachers need to differentiate between learners when they teach new content and skills to novices. This calls for teaching and learning strategies to be applied in order to differentiate within a large class of learners, but it remains a challenge to do so. Teachers should walk learners through the procedure and the concepts supporting the procedure, at least in the initial phases of explaining how to solve a mathematical problem. Subsequent exercises may then have little or no procedural explanations. Clark, Kirschner and Sweller (2012) interpret this partially guided approach as experiential learning or constructivist learning, among other synonyms. Experiential learning causes learners to experience the state of disequilibrium more frequently than the state of equilibrium during the process of equilibration.

According to Bormanaki and Khoshhal (2017), learning can be described, through pure discovery methods, as a state of discomfort or disequilibrium experienced by learners. Although discovery learning was used, research shows that students who mastered concepts using discovery learning “showed no signs of superior quality of learning” (Clark, Kirschner and Sweller, 2010:80). Above all the arguments put forward by the authors, the most important factor is the time it now takes in constructivist approaches, which may increase by days instead of a normal 45-minute period. This is why educators such as Peabody (2014) opt for rote learning.

The authors highlight an even more important concern in that learners will always choose the approach that has the least impact on their input to learn concepts. A less-skilled learner will rather opt for a less-guided approach for according to Clark, Kirschner and Sweller (2012) a more guided approach requires learners to provide a more attention-driven approach. Accommodation will increase the quality of the learner's schema and his/her thinking processes are challenged by greater adaptation. On the other hand, more-skilled learners will opt for a guided-approach for it requires less attention and thinking, but adds to the development of that learner through assimilation with minimum adaptation. A constructivist approach is regarded as a means by which students should construct their own knowledge. Many educators propagate this as the discovery of knowledge in solving problems without explicit guidance. This is also known as a "constructivist teaching fallacy" (Clark, Kirschner & Sweller, 2012:8). The authors further argue that hiding or withholding information from learners cannot help with the construction of knowledge. There is a difference between "constructing knowledge" and a "constructivist approach". The latter construct knowledge to a lesser degree.

The brain learns through long term, short-term or working memory (Mostyn, 2012; Sweller et al., 2019). Cognitive load theory can be applied to complex learning such as Mathematics and Science (Mavilidi & Zhong, 2019). Working memory can only hold information for a couple of seconds, unless the learner uses his/her long-term memory to fetch concepts previously learnt. Such as a chess player that can scan multiple chess board moves to make an informed best choice regarding the move, he/she needs to make. Long term memory thus provides a holding area for a "worked-example" in mathematics. The learner can then just like an expert retrieve the worked-example from long-term memory and successfully perform the procedure to solve a problem, based on that "worked-example". The authors also coined this as the "worked-example effect". They see novices spending a considerable amount of time engaging in problem-solving activities hardly learning anything during this engagement (Clark, Kirschner & Sweller, 2012). The aim of CLT is to "generate novel instructional techniques" (Sweller & Paas, 2017:86). CLT is regarded as the ability of learners to "process new information and construct knowledge in long-term memory (Sweller et al., 2019:262).

As early as 1928, Brownell (1935) advocated his "meaning theory". Brownell perceived meaningful learning as an important ingredient to successful learning. Meaningful learning, according to Brownell, can only take place when learners make sense of mathematics by understanding the concept and how it is applied in the real world. Higgins and Wiest (2006) posit that the slogan, "practice makes perfect", does not always hold true when an incorrect concept is practiced. Terms such as relative epistemology and constructivist teaching fallacy are commonly linked to concepts practiced, but not as per mathematical definition.

(v) Computational thinking

Problem solving skills and logical thinking, which encompasses abstraction and automation as its two main pillars, is computational thinking in a nutshell (Wing, 2006, 2008). Words such as “thinking at multiple levels of abstraction”, “decomposition”, “heuristic reasoning to discover a solution”, “prefetching and caching in anticipation of future use”, “recursive thinking” and “algorithm and precondition” describe some of the skills needed to think like a computer scientist (Wing, 2006:33).

Abstraction and automation are the “mental and metal tools” of computational thinking, respectively (Wing, 2008:3718, 2011). Denning (2017) regards computational thinking as being dependent on computational models, where computational models are abstractions as well. If computational models are absent for specific problems, computational thinking is the research activity for developing new computational models to secure solutions. The computational model is always paramount as a solution to a problem.

Computational thinking is a challenging process because it demands a high level of confidence, the ability to persist when one is confronted by complexity, the ability to deal with ambiguity, the ability to deal with open-ended problems, and the quality to work within a group in such a manner that the aims are achieved. After having mastered these important characteristics, the individual should be able to formulate problems in such a way that it can be solved tangibly through the use of computers or other tools. Data should be organised and analysed logically. Abstractions must be employed to represent data using models and simulations (Bar, Harrison & Conery, 2011). Algorithmic thinking should be applied, and all possible solutions need to be inferred by analysis and identification in order to achieve the most efficient and effective set of steps and resources. According to Denning (2017), these said steps or algorithmic design are needed to control some computational models. The ultimate solution should not be confined to that problem only, but need to be transferrable to other problem situations (Barr, Harrison & Conery, 2011). Computational thinking revolves around problem-solving skills, but it is applicable to almost any discipline (Wing, 2006). Wing further regards computational thinking as a fundamental skill for everyone, which is not particularly reserved for computer scientists, but this approach is challenged by Denning (2017).

Computational thinking encompasses quite a bevy of skills, but these skills are challenging to the average learner to understand and master. According to Philbin et al. (2013), problem decomposition, pattern recognition, pattern generalisation, defining abstract models, algorithmic design, and data analysis and visualisation are computational thinking techniques that need to be mastered.

Distinguishing computational thinking from the different thinking processes in mathematics and engineering is not easy (Cooper, Pérez & Rainey, 2010). Cooper, Pérez and Rainey (2010) developed a model for computational learning as opposed to traditional computational thinking, placing the emphasis on humans who use the computer and software to foster computational learning. More research is needed on the understanding of what computational thinking is and on strategies to assess ways in developing computational thinking by means of design-based learning activities (DBLAs) (Brennan & Resnick, 2012).

(vi) Cognitive ability and computational thinking

Cognitive ability refers to a number of keywords overlapping with the definition of computational thinking. Keywords used to define cognitive ability are the “cerebral function that allows one to acquire, memorise, recall, combine, compare, and use information and conceptual skills in new contexts” (Cegielski & Hall, 2006:74). According to the authors, while cognitive ability explains variability in task performance, the theoretical value belief and personality characteristics of an individual also plays an important role. Wing (2006) argues that computational thinking builds on computing processes, irrespective of whether the thinking processes stem from a human or a computer. Computational methods and models empower individuals to solve problems and design complex systems. Cognitive ability is an enabler of performing computational thinking. Computational thinking allows an individual to provide a solution by using abstraction and decomposition when designing large complex systems. Abstraction is one of the pillars of computational thinking (Wing, 2011) and will be discussed in the next section.

(vii) Abstraction in action

Researchers such as Hayakawa (1949), Truran (1992), Wilensky (1991), Dubinsky (1991), Hazzan (1999, 2003), Devlin (2003), Kramer (2007), and Perrenet (2010) state subtle differences when arguing the concept of abstraction. Hazzan (1999) views abstraction as a complex concept having many meanings for different contexts such as within mathematics. Kramer (2007) uses examples of abstract paintings in art to illustrate generalisation that focuses only on essential lines within a drawing. The author describes abstraction present in the map of the London Underground railway designed by Harry Beck in 1933. The map points to a concretisation of abstraction in a linear representation of the London Underground, which makes more sense to travellers, compared to a true representation of absolute routes and curves. This is a typical example where detail is removed, which differs from the concept definition of abstraction. Hazzan (1999) points out that a concept tends to be misinterpreted by students more often when students work at lower levels of abstraction than what is required. According to Perrenet (2010:89), “abstracting is the bringing to a higher aggregation level of a viewpoint (statement, model, theory) whereby it can be made

applicable to more cases". Others understand abstraction as extracting, drawing-out or 'abstrahering' commonalities from a set of objects (Meyer, 2010) by removing detail. This is also seen as performing a generalisation to identify a common focus point or essence (Kramer, 2007). The common denominator is the quality of the relationship of the individual with the object (Wilensky, 1991; Hazzan, 1999). Wilensky (1991:4) states that having the "right relationship" with the object can lead to a concrete conceptual understanding and "concreteness, then, is that property which measures the degree of our relatedness to the object (the richness of our presentations, interactions, connections with the object), how close we are to it, if you will the quality of our relationship with the object". Wilensky (1991) and Dubinsky (1991) regard abstraction as the degree of the relationship that exists between the subject (learner) and the object (concept definition). The degree of the relationship also points to the embodiment of the learner and the mathematics concept as researched by Bruner (1966) and Tall (2003, 2004).

In [Figure 2.7](#) below, learners are expected to abstrahere apples into numbers. The "appleness" (Meyer, 2010:2) principle is used by educators to teach learners for example to count.



Figure 2.7: Translate the "Appleness" of counting into a number object (Adapted from Meyer, 2010:2)

Tall (2008) describes this abstraction as proceptual symbolism. Learners commence with conceptual embodiment that leads to symbols or thinkable concepts called numbers such as 1+1 by performing an action schema called counting. Counting thus translates into a thinkable concept called numbers. Learners now move from embodiment to the symbolic world of procepts.

Learners are often given the freedom to solve problems in their own unique way and the teacher ticks boxes to confirm that the mathematical concept has been dealt with. In many cases learners may even get the right answer, but the method used to get to the solution is not always friendly towards learners, who may find themselves in a whirlpool of so many possible avenues to follow. This is when confusion dictates actions. Often these lower levels of abstraction complicate understanding, as Hazzan (1999) and Kramer (2007) emphasise in their research. This viewpoint of Hazzan (1999) and Kramer (2007) can also be considered for the South African context where too few learners are benefiting when abstraction becomes complicated. Researchers describe this state a learner is in as "abstraction anxiety" (Sfard, 1991; Wilensky, 1991; Meyer, 2010), which forms an important component of

mathematical anxiety. Papert (1980:8) uses the term “mathophobia” and Tall (2004:281) and Plerou (2014:XXVI) refer to this phenomenon as “dyscalculia”. Plerou (2014) not only categorises dyscalculia as the difficulty to understand number concepts, but also regards dyscalculia as a learning disability. According to Meyer (2010), educators should not only classify subjects as being abstract, but also deal with this anxiety associated with abstraction.

Many studies (Dubinsky, 1991, 2000; Hazzan, 1999; Wilensky, 1991; Dubinsky & McDonald, 2001; Kramer, 2007; Perrenet, 2010; Meyer, 2010; Maharaj, 2013; Brijlall & Maharaj, 2014) have been conducted on the concept of abstraction as a prerequisite for subjects such as Mathematics and Programming. However, many of these studies were conducted at university level where students already acquired the skill of abstraction within certain disciplines. Instruments for assessing certain characteristics associated with abstraction skills are then devised or used to measure abstraction (Hill et al., 2008; Perrenet, 2010). Unfortunately, only a few studies have been conducted at high school level from grade 8 to grade 12.

The next section investigates the type of programming language and computational thinking by discussing: (i) the positioning of a programming language; (ii) the characteristics of a programming language that satisfies Piaget’s learning; (iii) computational thinking and a programming language; (iv) Piaget’s cognitive level of formal operations and a programming language; (v) definition and choice of a programming language; (vi) traditional syntax-based programming languages are difficult to learn; (vii) programming language skills; and (viii) rote learning and embodied experiences in a programming language.

(b) Sub-research question (SRQ) 1.2

SRQ 1.2: What type of programming language may be used to promote computational thinking skills at a cognitive level of formal operations?

When performing coding at a cognitive level of formal operations, computational thinking is implied owing to thought processes that will govern coding (Aho, 2012; Selby & Woollard, 2014; Denning, 2017) when using a programming language at a cognitive level of formal operations (Piaget, 1964; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017). The term ‘computational thinking’ needs be revisited. How can learners be supported when they are involved in subjects that require computational thinking? In answering this question, major issues need to be identified, such as, “Where is computational thinking found in our educational space?” and “Can a programming language be used to invoke computational thinking among learners that may play important roles in this research to address the reason behind the lack of computational thinking skills among

learners in SA education? The next section will investigate computational thinking and the relationship with a programming language.

(i) Positioning of a programming language

Wright, Rich and Leatham (2012) identify the positioning of programming language teachings within technology education as an option, but do not prescribe where (in which grade) programming must be taught. The authors position the teaching of a programming language within the Standards for Technological Literacy (STL). The International Technology Engineering Educators Association (ITEEA) also does not prescribe an exact curriculum for technology teachings and hence suggests the positioning to be in STL. In South Africa, the researcher attended Oracle training on Greenfoot programming where a teacher attendee confirmed that she took the responsibility upon herself to teach learners programming skills from an earlier age than the prescribed level of Grade 10. There are more such cases at other schools, especially in private schools.

The fourth industrial revolution (4IR) is regarded as a disruptive technology that will generate jobs in developed countries of origin as opposed to the remote developing countries of consumption (Mpofu & Nicolaidis, 2019). The researcher partially agrees with the authors from a South African perspective, seeing that SA already embraced 4IR at educational institutions, with a strong DBE support. The researcher's stance is strengthened by how the South African government is dealing with the COVID-19 pandemic in 2020 and how the education departments are embracing programming and AI. However, if the youth of SA do not embrace the concepts of computational thinking, SA will remain the consumers of the developed countries. As correctly stated by Mpofu and Nicolaidis (2019:11), "the social effects of joblessness are devastating", but SA is a population with a magnitude of competencies that must be cherished to ensure prosperity for all its citizens and SA's strength lies in tapping into its diversity, which is lacking at the time of this writing. Szlávi and Zsakó (2017) strongly advocate ICT competencies as contained in the National Curriculum Statement of South Africa when considering Computer Science curricula.

(ii) Characteristics of a programming language that satisfies Piaget's learning

It seems that educators agree that programming must be taught earlier than Grade 10. The debate around which program to use is an on-going one (Papert, 1980; Papert, 2005; Cegielski & Hall, 2006; Kranch, 2010; Feurzeig & Papert, 2011; Lee, 2011; Brennan, 2012; Brennan & Resnick, 2012; Bule & Seith, 2012; Weintrop & Welinsky, 2018b; Szabo et al., 2019). A learner needs to be drawn into the programming language to discover a solution to a complex problem adhering to CLT. This eventually stimulates learning without forced

learning, called “Piaget learning” or constructionist learning, also known as “learning without being taught” (Papert, 1980:7). Brennan and Resnick (2012) rely on Scratch as a vehicle to explore the learning process using computational thinking. According to Brennan (2012) and Brennan and Resnick (2012), Scratch is a computational authoring tool developed by the Lifelong Kindergarten research group at the MIT Media Lab and the most popular programming language in education at the moment (Szabo et al., 2019). The authors of Scratch defend computational thinking being a primary goal within the teachings of Scratch. Scratch is a block-based language (Weintrop & Welinsky, 2018a) and requires no syntax, which makes it difficult to classify it as an OO programming language. However, Cegielski and Hall (2006) proposed a model of predictors for OO programming performance (Figure 2.8). The three enablers or predictors are (i) theoretical value belief, (ii) personality, and (iii) cognitive ability.

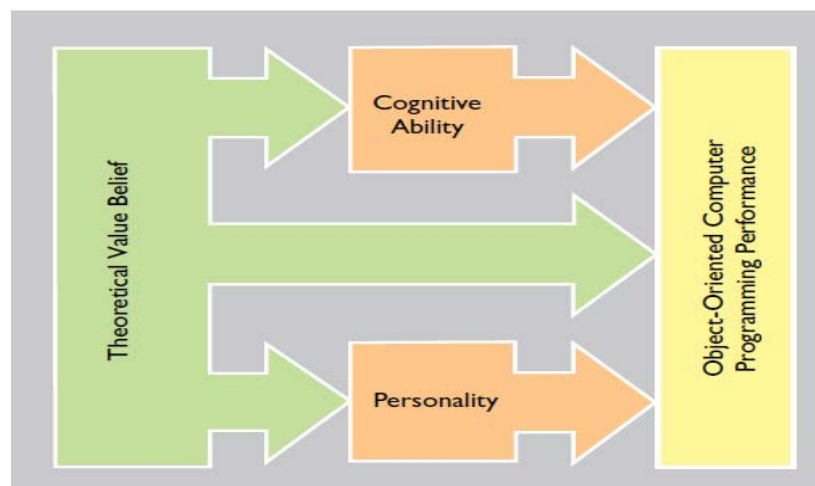


Figure 2.8: Model of predictors of OOP performance (Adopted from Cegielski & Hall, 2006:74)

The value beliefs as depicted in Figure 2.8 that motivate people in different ways are:

- i) Aesthetic value belief, motivated by art and beauty.
- ii) Economic value belief, motivate by pragmatism.
- iii) Theoretical value belief, motivated by the discovery of truth through personal standards like the need for order, problem solutions, and proof are all aspects of importance to the individual.

Value beliefs are the driving forces behind actions of the individual and hence theoretical value belief can be seen as the antecedent of personality and cognitive ability (Cegielski & Hall, 2006). According to Cegielski and Hall (2006:73), the usage of OOP in industry increased “dramatically”. Object-oriented PLs focus on the object and an object in a natural environment is three-dimensional, where Greenfoot is two-dimensional. Learning to program is a challenging activity and the question is whether the learner needs to climb the ladder of cognitive development before using Greenfoot. Many authors thus far described

computational thinking not only as the ability to decompose a problem into digestible steps or sub-problems, but as a holistic view evaluating the problem from different angles (Wing, 2006; Denning, 2009, 2017; Brennan & Resnick, 2012). According to Brennan and Resnick (2012), constructionism is achieved when learners use Scratch as a design tool in interactive media. The researcher opines that the statement of Brennan and Resnick opens up questions such as, “How does the researcher know this?” and “Can the degree of constructionism be measured?” The author further argues that constructionism refers to effective learning experiences that take place when learners are active in the construction of all things that are personally and socially meaningful (Papert 1980, 1993; Ackerman, 2001; Papavasopoulou et al., 2019). Learners interact with each other and reflect upon themselves or think about their own thinking to learn by design (LBD). Kolodner et al. (2003) found that learning science from design activities require multiple forms of support and learning opportunities, which points to using different tools to accomplish this. Sweller et al. (2019:276) describes this as “complex learning”. Brennan (2012) describes constructionism as designing, personalising, sharing and reflecting activities taking place when young learners use interactive media. Lee and Kolodner (2011) suggest two cycles for learning-by-design, consisting of design/ problem-solving/ redesign cycle and an investigation/ exploration cycle as depicted in Figure 2.9.

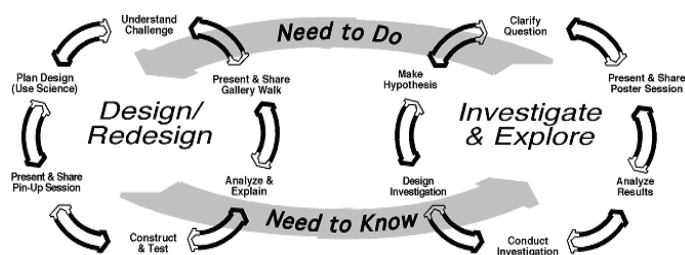


Figure 2.9: Learning-by-design cycles (Adopted from Lee & Kolodner, 2011:6)

The investigation/exploration cycle is entered into when the learner finds a need to learn something new. After learning has taken place, he/she returns to the design cycle to set new goals for learning to happen. This cycle motivates new goals and sub-goals (Lee & Kolodner, 2011). The authors further argue that any complex skill can only be acquired through practice and reflection on one’s own reasoning, which ties in with a term called ‘reflective abstraction’.

(iii) Computational thinking and a programming language

Data should be organised and analysed through abstractions to represent data by means models and simulations. Algorithmic thinking and all possible solutions inferred by analysis and identification to achieve the most efficient and effective set of steps and resources for a solution should be used. These steps or algorithmic design must control some computational models, as illustrated in Figure 2.10 (Denning, 2017).

The ultimate solution should not only be confined to that problem; it should also be transferrable or generalised to other problem situations (Barr, Harrison & Conery, 2011). Computational thinking revolves around problem-solving skills, but it is applicable to almost any discipline (Wing 2006). This concept of ‘computational thinking for all’ is disputed by Denning (2017) and others as pointed out in section 2.2.2, where a programming language is involved when computational thinking is researched.

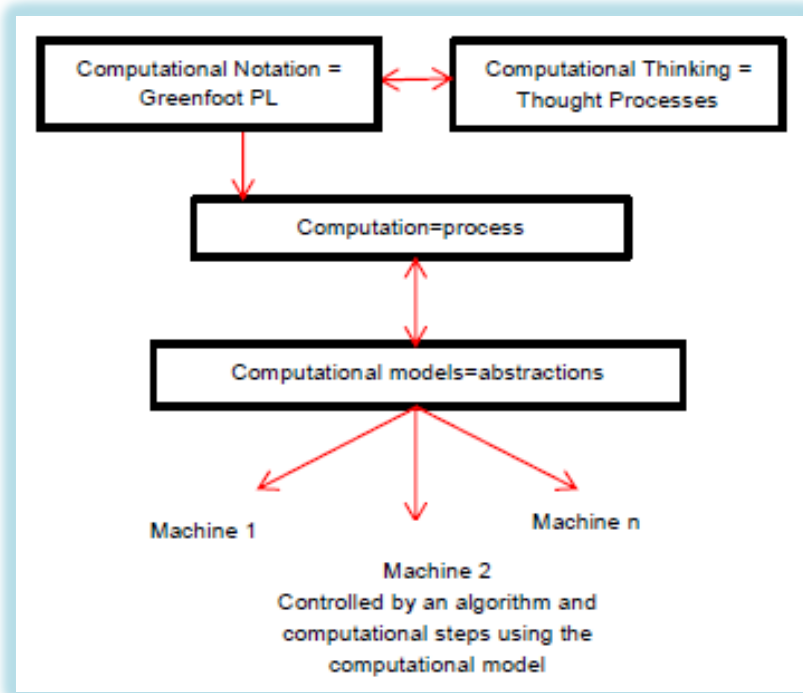


Figure 2.10: Computational thinking in motion (Adapted from Denning, 2017; Aho, 2011)

Computational thinking is a skill, and a skill can only be acquired over time, separate from the knowledge of facts (Denning, 2017). Competency is more than merely skillset development. In most cases, the learners are assessed on knowledge and not competency. Incompetent learners then end up in industry, which is highlighted by industry, when receiving students for experiential learning during their academic studies (Selby & Woollard, 2014). This is analogous to the term “low road transfer of knowledge”, stated by Wilhelm (2008:45). Denning (2017) posits that competency-based learning must enjoy greater attention within an academic programme for the learner to fully embrace computational thinking. These competency assessments must be given guidelines as to address different skill levels within computational thinking. Following any sequence of steps or algorithm does not necessarily make someone a computational thinker (Denning, 2017). Computational thinking is about finding appropriate models of computation in order to find a solution for a formulated problem (Aho, 2011). Linking APOS theory with computational thinking, reflective abstraction is used in the context of computational thinking (discussed in section

2.2.2.1(c)(v) (Cetin & Dubinsky, 2017). Students in urban school districts are only taught basic literacy skills and subsequently fail to realise the huge computer technology investment possible, all because of a lack in computational thinking education (Pearson, 2008).

(iv) Piaget’s cognitive level of formal operations and programming languages (PLs)

Many students fail their first programming course because of the frail mathematical foundation provided, but this degree and type of algebra course necessary to improve programming skills can be a challenging one, because of the complex nature of a student profile. Piaget’s cognitive theory should be considered when dealing with prerequisites of mastering PLs, and Louden (1993) specifically highlights procedural programming (White, 2003). Cegielski and Hall (2006) and Barr, Harrison and Conery (2011) posit that cognitive ability is not the only enabler for abstraction. There are three levels of cognitive development, known as pre-operational, concrete, and formal operations. It is at the cognitive level of formal operations where learners from the age of 12 years onto adulthood will be able to think more abstract, show deductive logic, and understand conceptual thoughts (Cherry, 2014). White and Sivitanides (2002) classify PLs (Figure 2.11) into a specific level of cognitive development (Piaget, 1977; Papert, 1980).

Table 1. Programming Languages and Cognitive Development/Style

Programming Paradigm	Piaget’s Cognitive Development Levels				Cognitive Style (Hemisphericity)
	Pre-Oper	Concrete	Pre-Formal	Formal	
Procedural (COBOL, logic sequence)	Burnout	Burnout	Burnout	P & M	Left Brain
Object Oriented (C++, Java, concepts)	Burnout	Burnout	Burnout	P & M	Either Hemisphere
Visual (Visual Basic, on screen)	Burnout	Burnout	P & M	Bored	Either Hemisphere
Script (HTML, Web Pages)	Burnout	P & M	Bored	Bored	Right Brain

P & M: Productive and Motivated

Figure 2.11: Cognitive development style of PLs (Adopted from White & Sivitanides, 2002:63)

Object-oriented programming (OOP) in Figure 2.11 shows that OOP it is a productive and motivated activity at a cognitive level of formal operations. Although a huge body of research is available on mathematics as the paramount contributor of being successful at programming, Portnoff (2018) states that natural language may be an important role player when looking at neurocognitive evidence during CS programming education. This research also regards the way a learner constructs an algorithm as natural language dependent.

(v) Definition and choice of a programming language (PL)

Traditional PLs are best understood by studying several PLs available (Organick et al., 1989). Programming is the act of understanding a problem, formulating a solution, and writing down the solution in such a way that a computer can use the solution to solve the problem (Moström, 2011; Aho, 2012; Denning, 2017).

Programming also evolved to accommodate the language of your choice. A common spoken language does not suit the purpose because of its ambiguity. This viewpoint however changed recently when Felleisen et al. (2018) stated that PLs changed to become domain specific and language-oriented through specific interfaces. According to Felleisen et al. (2018), a programming language is a problem-solving tool in itself. The authors regard a programming language as an abstraction built around integrity. The authors further argue that object-oriented programming and concurrency-oriented programming revolutionised the world by utilising objects and message passing techniques. Software is now developed in different ways and most research studies are sparked from these innovations. However, Felleisen et al. (2018) advocate the use of embedded domain-specific languages (eDSL) that may have a stronger relationship with the actual definition of computational thinking, namely that it is for every discipline and not for Computer Science only. Research is done on a programming language for programmers, or a language-oriented programming (LOP), with a new syntax as well as static and dynamic semantics that map the new syntax to a host language and peripheral languages by using a foreign-function interface instead of the normal integrated development environments (IDEs). However, programming research does not yet provide software developers with the tools formulate solutions in the languages of problem domains (Felleisen et al., 2018).

The challenge learners now have to face, with available tools, is to abide by strict syntax to accomplish an algorithm using a programming language. Some, but not all, authors restrict the term 'programming language' to those languages that can express all possible algorithms. Once again, 'express' can be accomplished in many ways. Seeing that this research refers to a learner aged 12 and older, the choice of a programming language revolves around Scratch, Alice and Greenfoot, whereby usage of syntax is minimised. The DBE (2011) regards Scratch as a fun tool to assist learners with acquiring computational thinking and programming skills through algorithmic design. Greenfoot expects of learners to demonstrate more competence compared to Scratch (Maloney et al., 2008; Kölling, 2010b). Greenfoot entails applying object-oriented concepts in applications and the ability to scale; furthermore, Greenfoot it is easier to start with, compared to Scratch and Alice for first time users.

Scratch is a programming language with the ability to create sophisticated computer programs by selecting visual programming blocks using a pointing device. No typing or writing of syntax statements for programming constructs is necessary, only the use of programming blocks. These programming blocks will only fit together when the syntax is correct (Lee, 2011). Language plays an important role in mathematics (Tall, 2008), and hence the researcher excluded Scratch from the list because of the over-emphasis on the visual approach only (Boyle et al., 2003). Perceptions of Scratch programming among learners are determined by the exposure of these learners to programming. Although it is argued that Scratch is intended to increase computational thinking through a cognitive building-block approach, the actual programming part is neglected, which then prevents learners from expressing themselves using a programming language with syntax and not having to write text commands (Marimuthu & Govender, 2018:58). This perception that Scratch does not assist with transitioning to a real programming language like Java when the focus becomes syntax, is also conveyed by Grade 11 learners exposed to the Java programming language. Brennan and Resnick (2012) defend the usage of Scratch as the way learning takes place, as opposed to teaching Scratch as a programming language. Scratch forms part of the block-based visual programming languages. Papavlasopoulou, Giannkos and Jaccheri (2019) also describe Scratch as a block-based visual programming environment. Novice programming environments (NPE) such as Scratch allow both learners and teachers to gain entry into the world of programming (Papadakis et al., 2014). Perhaps the word 'language' should be omitted from the description and replaced with programming environment (PE). To consider the way learning takes place is important, but it places more emphasis on the teaching style and expertise of the teacher than on the programming language. The programming language should provide deeper learning and scalability into language constructs, with near and positive learning transfers (Wilhelm, 2008).

Alice, a three-dimensional sister of Karel, used for algorithmic thinking, is another block-based visual programming environment. It is "primarily a scripting and prototyping environment that allows the user to build virtual worlds and write simple programs to animate objects" (Zsakó & Szlávi, 2012:57). Alice is also an OO three-dimensional, interactive environment without the requirement of explicit syntax (Ebrahimi, Geranzeli & Shokouhi, 2013). Alice is successful owing to its drag-and-drop programming interface and its "rich and engaging 3D environment" (Cohen, 2013:82). Alice 3 allows the learner to import Alice programs into NetBeans and extend the development of programs using industry standard tools and programming languages. The author further states that a development path exists into Eclipse IDE, which emphasises scalability (Cohen, 2013). Alice is also introduced as a programming language by the Oracle Academy, forming a progression path with integration capabilities for learners who choose Information Technology as a subject in Grade 10. Alice

makes use of property tables, but the actual language used for logical reasoning is hidden. Another motivation in choosing a programming language for this research, according to Papert (2005), is that children can use computational ideas to increase the understanding of their own thinking and learning. The progression path is overshadowed by Papert's (1980) constructionist approach, with emphasis on an instant result as opposed to traditional preparation of a programmer. Although Papert is referring to Logo, the language of his choice that satisfies instantaneous results, Greenfoot is also a programming language or tool that achieves just that for the majority of learners within any society. The wide range (rich set) of applicability that Greenfoot provides and which aligns with higher order thinking and usage, also coincides with the theory of Papert (2005). Papert argues that educationists teach learners many disjoint aspects, such as grammar, history and numbers, with the hope that something important will happen for these learners to make sense of a disordered environment of topics and facts. The researcher regards this statement of Papert as an important consideration when using a programming language to teach, if computational thinking cannot be clearly identified within teachings of a novice programming environment (NPE). Greenfoot contains a combination of graphical actors and scenarios available to learners. The learner uses the scenarios and actors to generate objects that satisfy a selected algorithm. The combination of these objects with the algorithm creates an absolute reality. Owing to the variation of Greenfoot to solve problems using a graphical user interface, learners may engage in the Java programming language to achieve similar outcomes. This is mainly because the Java Development Kit (JDK) is needed as prior installation of Greenfoot programming language. Having this progression path to Java, learners are able to express themselves through language as well. The language may vary between Greenfoot programming language and Java syntax used within the Greenfoot IDE, thereby not limiting learners who need differentiation in difficulty levels when using the Greenfoot programming language. As opposed to visual constructs in Alice and block-based Scratch, Greenfoot was chosen as the programming language with its embodied approach, scalability, optional language statement approach and OO application.

(vi) Traditional syntax-based PLs are difficult to learn

Having discussed the variety of visual and block-based NPEs such as Scratch, Alice and Greenfoot, the pros and cons of traditional PLs need to be investigated. Kranch (2010) states that PLs consist of syntax and semantics, and students need to investigate syntax as a starting point. Educating students in PLs are filled with challenges, and what seems to have been a "grand" challenge twelve years ago, still applies (Bennedsen & Caspersen, 2019:35). According to Robins, Rountree and Rountree (2003), programming courses are difficult and the question of what actions need to be implemented to make a good programmer is an ongoing debate since 1970. The authors further argue that problem-solving and

computational thinking lack among students who take introductory programming courses that use a text-based programming language. On the other hand, McGettrick et al. (2005) argue that introductory programming courses are needed to address the problems of high dropout rates and negative attitudes towards a programming language. The introductory programming courses should start with some NPE as a gradual introduction, before venturing into syntax-based PLs.

Concepts such as rigorous thinking, variables, classes, decomposition, debugging and generalisation require a better or perfect understanding of the programming language and algorithms (Malan & Leitner, 2007; Saeli et al., 2011) because of the preciseness of programming language syntax expected from a programmer. Programming is a skill that requires other skills (Rahmat et al., 2012).

Saeli et al. (2011:80) reported the following complexities while studying traditional PLs:

- Orientation (benefits of learning to program)
- Notional machine (understand the hardware one needs to control)
- Notation (syntax and semantics of a programming language)
- Structures (e.g. loops to achieve goals)
- Mastering the pragmatics of programming, i.e. learning the skills to specify, develop, test and debug

There is a need for object-oriented programming languages (OOPLs) that can be addressed using visual programming languages such as Greenfoot (Marimuthu & Govender, 2018). Using and OOPL is a fun way to learn programming and remove the fears learners have about a programming language in general as opposed to traditional PLs (Meerbaum-Salant, Armoni & Ben-Ari, 2013).

(vii) Programming language skills

The usage of exact syntax to create a computational model may be difficult for the majority of learners (Malan & Leitner, 2007; Saeli et al., 2011). The process of mapping an algorithm using a programming language must be done in an unambiguous way without any mistakes (Papert, 1980). Algorithmic thinking is a type of mathematical thinking simulating “intuitive pattern recognition and analogical thinking” (Zsakó & Szlávi, 2012:58). The authors further argue that programming includes specific skills such as:

- i) functional decomposition,
- ii) repetition, which may include iteration and/or recursion,
- iii) basic data organisation such as lists and arrays,
- iv) generalisation and parameterisation,
- v) algorithm versus program,

- vi) top down design, and
- vii) refining a design.

Learning any programming language may be a paradigm shift in that it is problematic to shift from procedural to object-oriented programming, but not *vice versa*. Exposing students to a paradigm shift where students are forced to learn more than one programming language belonging to a different paradigm, such as procedural and object-oriented PLs, may thus create more complexities in terms of understanding. Many students in general are not capable of dealing with the challenge because of their cognitive level of development (Mazaitis, 1993; Kölling, 1999a, 1999b). Declue (1996) and Gomes and Mendes (2007) support the views of Kölling (1999a, 1999b) and Mazaitis (1993) and state that learning to program requires intensive practice. This statement has been relaxed since the inception of novice programming environments (NPEs).

Problem solving skills of novice programmers is a major hurdle (Saedi et al., 2011). Weigend (2006) argues that novices generate wrong outcomes. Weigend (2006) sees a challenge with achieving the semantics when learners have to map an algorithm into a programming language; although they (the learners) know the intuitive solution, they are not able to produce the code for that intuitive solution. The focus here is not only syntactically but also logically.

Notwithstanding all the disadvantages, the advantages of acquiring the skills to program are pointed out by Papert (1980, 2005), who shows that the learner may investigate aspects of science, mathematics and the art of intellectual model building, when practicing programming. Papert (2005) also argues that programming creates experiences for learners to strengthen their intuitions and concepts on how thinking, learning and playing take place. Szlávi and Zsakó (2017) view language abstraction, analogy, algorithmic abstraction, decomposition and superposition, conversion, intuition and variation as a cognitive toolkit for programming where cognitive operations take place in the conscious and subconscious mind.

(viii) Rote learning and embodied experiences in a programming language

According to Bruner's (1966) mathematical research in [Figure 2.4](#), mathematics consists of enactment, iconic and symbolic representations. PLs may provide all these properties to guide a learner into the do-able process, creating thinkable concepts from these do-able processes. The programming language Greenfoot is rich in procepts, drawing on enactment, i.e., iconic representation through symbols. However, rote learning may become a "way of life" (Tall, 2004:286) if the learner cannot turn specific processes into thinkable concepts, as

in the case of mathematics as opposed to using a programming language that supports many solutions to one problem.

(c) Sub-research question (SRQ) 1.3

SRQ 1.3: What constructs within the programming language facilitate APOS theory at a cognitive level of formal operations?

The research method adopted is a literature analysis to cast more light on SRQ 1.3. Concepts such as computational models, computational thinking, computational notation and algorithms are researched, as well as how they interact to create abstractions when a computational notation such as Greenfoot is used (Figure 2.11).

In order to show dependencies among a programming language and APOS theory in promoting computational thinking, APOS theory is discussed in (i) foundational research towards APOS, (ii) didactical situation when teaching mathematics, (iii) how APOS promotes computational thinking, (iv) literacy, (v) abstraction, (vi) APOS theory and the Greenfoot programming language, and (vii) the role of mental mechanisms in mental constructs within Greenfoot.

(i) Foundational research towards APOS theory

Fischbein (1987) researched fundamental intuitions, the algorithms that promote computation and symbolic manipulation, and formal axioms for definitions and formal proof. Skemp (1971, 1979) researched human learning in the classification of humans having receptors to receive information, effectors to act on what they receive, thereby creating a system on which they impose operations and forever reflect on the system. The learners thus undergo three types of activity, namely (i) perception or input, (ii) action or output, and (iii) reflection, which entails perception and actions all over again, climbing the ladder of cognitive development over time. Bruner (1966) highlights the complex world of mathematics by distinguishing three modes of mental representation, namely, the sensorimotor, the iconic and the symbolic mode.

Revisiting SRQ 1.1, Tall (2004:282) coins his view as the “three worlds of mathematics” being the perceptions of the world around the individual to perceive and sense the mental and physical world. The world of symbols is described as where the individual switches between the processes of doing mathematics, to thinking about mathematical concepts, to the third world called the formal or formal axiomatic world. The journey of each individual may be influenced by “met-befores”, which may force the individual into a relative epistemology (Tall, 2004:286). This is a challenge in the learning process of any learner, as many learners develop a belief system about mathematics (Moscucci, 2007; Moscucci &

Bibbo, 2015) that every operation on numbers does have an answer. This is not the case when symbols are intermixed with integer values in algebra and there is no definite answer, such as $3x + 7$, where x may have a varying value.

As depicted in Figure 2.12, the focus is on embodiment and symbolism. Tall (2008) highlights learners' approaches as varying and put his three worlds forward as a theory that provides a framework by which mathematical learning and thinking at all levels of schooling can be considered.

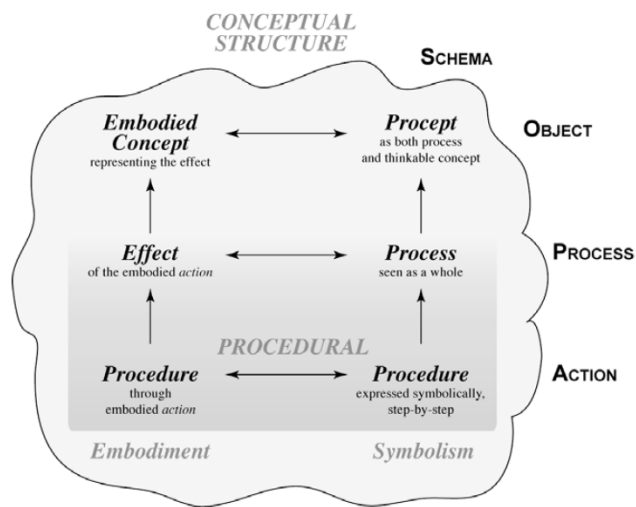


Figure 2.12: Procedural and conceptual knowledge (Adopted from Tall, 2008:12)

Tall (2008) illustrates in Figure 2.12 the impact of the APOS theory compared to his procept conception. Tall discusses many practical ways in which mathematics concepts can be explained using practical solutions as depicted in Figure 2.13. Determining the sum of the inner angles of a triangle, the corners of the triangle are torn off and placed adjacent to one another, which resembles a straight line and equals 180 degrees. This requires actions such as tearing of the paper and physically arranging the torn pieces of paper on a ruler.

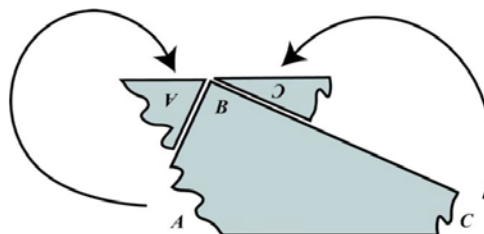


Figure 2.13: Tearing corners of a triangle to form a straight line (Adopted from Tall, 2002:9)

Tall's research on the three worlds of mathematics also provides insight into the importance of embodiment and how learners need embodiment as part of their learning until reaching

the formal axiomatic level progressing through the three worlds. The learner is submerged into a procept by thinking about the concept, such as the sum of the inner angles of a triangle, and by adding the embodiment to the process by involving the learner through these actions; the mathematical concept is better understood. APOS theory on the other hand uses the word 'Action', whereby embodiment of a process points to the involvement of the learner taking action, which then develops into a Process within the learner's mind, not having to take the same action again such as tearing off the corners of a triangle and place it on a straight line. This process then develops into an Object around basic facts of a triangle in geometry, which will be added into the Euclidean geometry schemata of the learner.

In one of Tall's (2008) papers, a hypothesised learning framework (Figure 2.14) was suggested for possible routes by which the teacher would guide the learner to cope with the complexity of the derivative. Unfortunately for the researcher, the role of the teacher and the role of the learner are illustrated as a framework and not in the form of a model. The same applies to Tall's three worlds, also discussed as a framework, which does not provide much information on the actual role of the teacher or how the learner will apply his/her mind. Tall does provide in-depth discussions on the actual mathematical concept, such as the cognitive development of proof and how this is dealt with, within the three worlds. Although the actual mathematical concept is discussed in detail, it ignores the processes of what might have happened in the mind of the learner and teacher. The framework is quite clear, but the detailed aspects of how the embodied and symbolic world will operate, is left for further research. The framework in Figure 2.14 also states that the teacher is the responsible mentor to guide the learners in a variety of ways to fuse knowledge structures that will make sense (Tall, 2008). The role of the teacher in how his or her responsibility must be carried out towards implementing mathematical concepts is absent from the framework. In South Africa, the DBE already implemented mathematical workshops for teachers. The teachers may also have instilled a relative system of belief about mathematics within these learners (Moscucci, 2007). If teachers are not explicit about the 'how', then leaving it to teachers to guide learners may be a challenge within South African education.

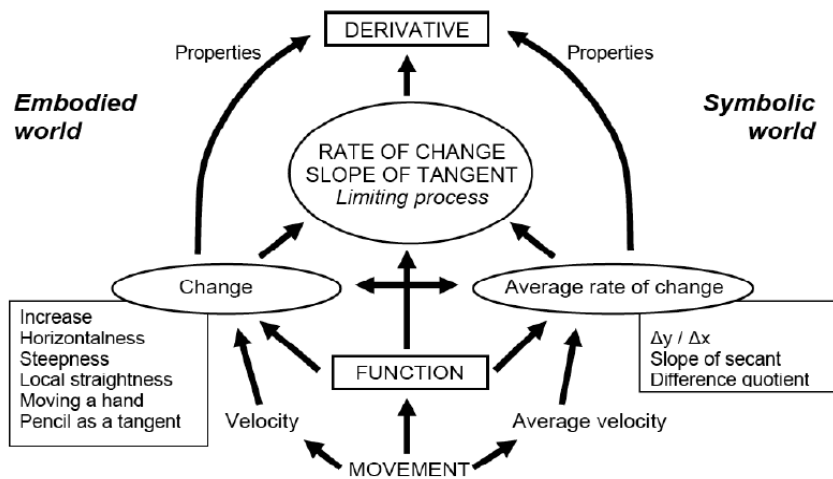


Figure 2.14: A learning framework (Adopted from Tall, 2008:14)

APOS, on the other hand, is a theory, framework and model (Arnon et al., 2014). Research in many articles shows exactly how the APOS theory is rolled out as a model. Cetin and Dubinsky (2017:72) regard mathematics learning vested in four aspects, namely: (i) the APOS theory; (ii) the ACE teaching cycle; (iii) computational thinking; and (iv) reflective abstraction. The authors further argue that APOS theory is renowned for the discovery of certain mental constructions that may lack among students, as pointed out by the majority of research done in the APOS field. A possible solution for this situation may be found in the APOS theory (Arnon et al., 2014). APOS is the acronym Actions, Processes, Objects and Schemas, which are mental structures. APOS is a constructionist theory, based on mental mechanisms or reflective abstractions such as interiorisation, encapsulation, de-encapsulation, coordination, reversal, generalisation and thematisation. Reflective abstraction is used in context of computational thinking (Dubinsky, 1991; Arnon et al., 2014; Cetin & Dubinsky, 2017).

Pedagogical strategies such as the **A**ctivities, **C**lassroom **D**iscussions and **E**xercises (ACE) Teaching Cycle may assist students and teachers with the understanding through the mental constructions on a specific mathematical concept, known as a genetic decomposition (GD). The embodiment within Tall's (2002:9) research will be ideas that teachers can bring, such as illustrated in Figure 2.13 during mathematics teaching, but it is not sustainable in all cases and depends on the creativity of the teacher. The tearing of the paper triangle does not provide extended use to provide clarity on other mathematical concepts, compared to the versatility of a programming language. The aim is to promote computational thinking skills at a cognitive level of formal operations among high school learners in a structured manner.

(ii) Didactical situation when teaching mathematics

Learners are instructed to correct or reproduce a solution similar to a mathematical problem, but with a different example. Because of a similarity with the new example, learners may get it right. So, both teacher and learner may think that they achieved success in ensuring that the learner understands the problem. The learner only interpreted a didactical intention, which is epistemological fraud (D'Amore, 2008). D'Amore (2008:11) regards epistemological fraud as the learner reproducing the same solution method for a problem, based on establishing a similarity with another exercise, which was done for the learner by another individual. Brousseau (2002:225) coined the didactical contract as when "specific habits of the teacher are expected by the student and the behaviour of the student is expected by the teacher". The "Jourdain effect" or fundamental misunderstanding is where both teacher and learner feel comfortable with the result that does not contribute to learning a mathematical concept per definition or even understanding that mathematical concept definition. From the didactical contract, phenomena such as the "Topaze effect" and the "Jourdain Effect" emerged (Brousseau, 2002:265). The "Topaze effect" is when the teacher wants to engage learners in a manner where the answers are hidden. The "Jourdain effect" is when the teacher guides the learner, motivating the learner in having enough or appropriate knowledge to solve the problem. Didactical situations can be a-didactical where the teacher uses a situation, known as devolution, when the learner is provoked by the teacher to feel responsible for performing academically well.

Tramonti, Paneva-Marinova Pavlov (2017) proposed a framework based on Brousseau's (2002) three types of didactic situations, namely:

- i) A-didactical – Learners learn mathematical concepts through discovery and not through rote learning.
- ii) Didactical – Learners learn mathematical concepts through teacher involvement, where the learner is guided from concrete, to pictorial, to the abstract phase.
- iii) Non-didactical – Teachers will act as mediator, but allow students free use of their creative skills when producing artworks on math concepts.

The next section highlights APOS and computational thinking.

(iii) How does APOS theory inform computational thinking?

The APOS theory, framework and model originated from Dubinsky's (1991) interpretation of Piaget's (1973, 1975, 1977) concept of reflective abstraction. Piaget sees the properties of objects, not in the objects itself, but the properties are embedded in the actions that learners take when using these objects (Arnon et al., 2014).

Each mental construction or conception uses mental mechanisms (interiorisation, coordination, reversal, encapsulation and thematisation) to move through the APOS (mental structures) cycle. Reflective abstraction is a possible description of what goes on in the minds of individuals when engaged in creating knowledge, which is hypothetical, for nobody can see what goes on inside another's mind (Dubinsky, 1991, 2000). The next part of this research focuses on Piaget's contributions and how reflective abstraction fits into APOS. The literature review covers literacy, abstraction, and measuring abstraction.

(iv) Literacy

Prensky (2008) argues that programming needs to be regarded as a new form of literacy and the current 3R's of "Reading, wRiting, aRithmetic" must be expanded to pRogramming as well. Wright, Rich and Leatham (2012:5) argue that being literate, whether reading literate or programming literate, means to "manipulate the object of literacy, e.g. "communication technologies, in a meaningful way". They draw an analogy that being able to read, does not imply being literate. Wing (2011) also regards computational thinking as part of being literate.

(v) Abstraction

Many authors have debated abstraction for decades, but few brought a practical implementation of abstraction to the table for subjects such as mathematics and programming. An analysis on abstraction led to research on mathematics by researchers such as Skemp (1976), Dubinsky and Lewin (1986), Dubinsky (1991, 2000), Tall (2008), Hazzan (1999) and Kramer (2007). Certain researchers' views are now highlighted.

◆ Hazzan and abstraction

➤ Quality of the relationship between the object of thought and the thinking person

Anything being abstract or concrete must be seen as a property of the relationship between the person and the object. The more connections that person forms to the object, the more concrete he/she feels to it (Wilensky, 1991). Hazzan (1999) states that new knowledge can only be constructed based on existing knowledge. This is similar to Tall's (2008) "met-befores" concept. This also applies to objects being constructed based on existing mental structures. The more a learner becomes involved with an unfamiliar concept, the more he/she engages in a process that ends in conceiving the concept as an object.

➤ Reflection of the process-object duality

Hazzan (1999) states that educators need to distinguish between process conception and object conception when evaluating the duality of these concepts, also coined by Piaget as reflective abstraction. Processes are separated from content, and the processes are converted to objects of content. Reducing abstraction refers to a lower level of abstraction

when a mathematical concept is interpreted, also known as process conception. Process conception becomes personal when students express themselves using first-person language, e.g. “I want to check if I can find” It reflects a feeling that the person is doing something himself/herself (Hazzan, 1999:80).

➤ **Degree of complexity of the concept of thought**

Hazzan (1999) sees a compound object as more abstract. It is then expected that the student reduce the abstraction level in order to deal with compound objects not yet constructed as a mental object. The learner should deal with whatever abstraction he/ she faces, instead of reducing it for the learner. Specialisation refers to the identification of a smaller set or even one object within the larger compound set of objects in order to analyse a specific case (Cetin & Dubinsky, 2017). This may guide the student towards a general solution, but if mental structures are not constructed to deal with a general case, students may not reach that generalised solution (Hazzan, 1999).

◆ **Kramer and abstraction**

Abstraction is a basic principle in software engineering, if complexity should be mastered. Able students and good lecturers are those with an ability to perform abstract thinking and have abstraction skills (Kramer, 2007).

➤ **Is abstraction about removing detail?**

Kramer (2007) refers to abstraction as removing detail in order to simplify things. Kramer also uses the term “generalisation” to identify a common focal point or essence; this term can be compared to Piaget’s empirical abstraction, which does not always hold true for every situation. The author extracts common features from specific examples, but a specific circumstance and may not be transferrable in all circumstances. Kramer (2007) also supports the concept of abstraction as put forward by Hazzan (1999) and Cetin and Dubinsky (2017) in that abstraction has different meanings for different concepts, e.g. The London Underground map of 1928 and 1933. This map illustrates that it can only be used in context of train commuters, and not by taxi drivers. In this specific case, the generalisation of the map will not suffice for taxi drivers. Kramer (2007) further argues that abstraction is necessary for Computer Science and Software Engineering. Computing is all about constructing, manipulating and reasoning about abstractions. Writing programs is similar to handling abstractions in a precise manner (Devlin, 2003). Kramer (2007) describes abstract interpretation for program analysis as the generalisation of abstraction in programming. Another generalisation example is the usage of data abstractions and classes in OOP.

◆ **Arnon and abstraction**

Abstraction can be classified into empirical, pseudo-empirical and reflective abstraction. Each type will now be discussed to show the relevance of reflective abstraction in this research.

➤ **Empirical abstraction**

Empirical abstraction is where knowledge is derived from the properties of objects. These properties seem external to the learner or person, but the knowledge of the properties is the outcome of constructions made internally regarding interventions undertaken by the learner or person. The interaction happens between the learner and the object's properties (Arnon et al., 2014). Empirical abstraction is synonymous with the concept of extraction (Cetin & Dubinsky, 2017).

➤ **Pseudo-empirical abstraction**

Pseudo-empirical abstraction is when the learner creates relationships through internal constructions amongst the properties of the objects. According to research done by Dubinsky (1991), Piaget (1977) created the term to denote an intermediate stage between empirical and reflective abstraction. According to Dubinsky (1991), it is within this phase that knowledge is generated on the properties of an object by the learner, owing to actions introduced or taken by the learner. If the learner engages in thought on these properties of an object, it promotes the quality of the relationship between the learner and the object. The learner makes internal constructions with the objects by gaining more knowledge about the objects. This knowledge that the learner develops of the objects, strengthens the relationship the learner has with the objects. Euclidian mathematics is based on nine theorems. Only when learners interact with problems by interacting with the object in thought, can a relationship be built (Arnon et al., 2014).

➤ **Reflective abstraction**

Reflective abstraction, according to Dubinsky's interpretation from Piaget's research, is when actions now emerge from within the learner, so that these actions are coordinated in general and internalised to form a whole/new understanding which originated from these new actions coming from within. The actions are not attributed to external stimuli, but from mental processes within the mind. This can only happen if the learner considers what s/he already knows, and these actions can now be regarded as an empirical abstraction, without considering external objects anymore. It is thus a "new synthesis in midst of which particular laws acquire new meaning" (Piaget & Garcia, 1989:299; Dubinsky, 1991:97). It is a case of empirical abstraction all over again, but now it is an assimilation of schemas created during reflective abstraction (Dubinsky, 1991). Assimilation is a Piagetian term, meaning to add

knowledge to existing schemas in the brain, but not changing the structure of what is inside the brain. Accommodation happens when knowledge is assimilated into the brain, but changes should take place within the brain to create a meaningful and useful construct of this new knowledge (Bormanaki & Khoshhal, 2017). Assimilation can be prevented when the new knowledge or mental construct does is not congruent with the current structure or schema. The forced accommodation schema must be created instead, through equilibration. An analogy would be when learning takes place and new actions bring about new knowledge and understanding.

Dubinsky refers to these internal objects, using Piaget's term, as cognitive structures being formed. It also appears that Piaget formed generalised concepts regarding individuals. External objects, i.e. programming languages in its visual format, form a new variable to his theories in that individual development takes on priority above the development of all learners applying one theory about their development. Dubinsky (1991) states clearly the author does not suggest that interiorising advanced mathematics is done by applying reflective abstraction. Reflective abstraction is just a description of the process that happens within an individual when intellectual thought develops. Cetin and Dubinsky (2017:72) describe reflective abstraction as “reflecting on operations on a lower level and reconstructing and integrating them on a higher level”.

Reflective abstraction is not about drawing out common features, but on ways of acting on things. It is about the operations that individuals perform on objects and not about the properties of those objects without the interactions of the individuals in relation to an object; the object remains an object. A schema is regarded as those mental structures in the mind of the learner who understands or develops an understanding of a concept (Arnon et al., 2014). A schema may also be thematised into an object on which actions can be performed to make it part of another schema(s). The most important part of a schema is that it is a coherent set of objects and actions used by the learner to perform on these objects. A concept image, on the other hand, can be seen as a set of mental images in the mind of the learner associated with the name of the concept. The concept image is based on the learner's experience of the concept and not on the definition of the concept (Arnon et al., 2014). One can thus safely say that a schema of a concept is not the same as a mental image of a concept.

◆ **Measuring abstraction**

A measuring tool to measure the degree of abstraction of a learner is still to be found (Kramer, 2007). Although Perrenet (2010) claims that he discovered a measure, Dubinsky (1991) states that it is not possible to know how learners or subjects construct concepts; reflective abstraction can only help us to understand a concept, if and only if reflective

abstraction is considered as part of a study, observing a learner's acquisition of a mathematical concept. This is possible through observing and intervening as the learner investigates mathematical concepts or programming language concepts. The learner should develop a general approach to discover and interiorise future concepts through administering the general theory of reflective abstraction.

◆ **Genetic decomposition**

Dubinsky (1991) uses a term called ‘genetic decomposition’ (GD), which denotes a description, in using his general theory based on empirical data, as extracted by studying the learners as they create mental constructions of concepts in mathematics to further their understanding of a concept. The analysis consists of a synthesis on how an aspect within the programming language or mathematical concept can be learnt, based on an intervention by a teacher. Dubinsky (1991) identified three types of abstractions that were put forward by Piaget (1977) during his research, namely, empirical abstraction, pseudo-empirical abstraction, and reflective abstraction as discussed above.

(vi) **APOS Theory and Greenfoot**

◆ **Mental constructions and Greenfoot**

The mental constructions, Action, Process, Object and Schema (APOS), need to be identified within a programming language. The criticism is that mathematics is abstract already, but so is a programming language. The learner takes mathematics since his/her first day at school, and introducing a programming language at a later stage in his/her life, creates a superset of schemas within his/her intellectual thought, thereby accommodating not only programming language schemas, but also mathematical schemas, and definitely other schemas as well.

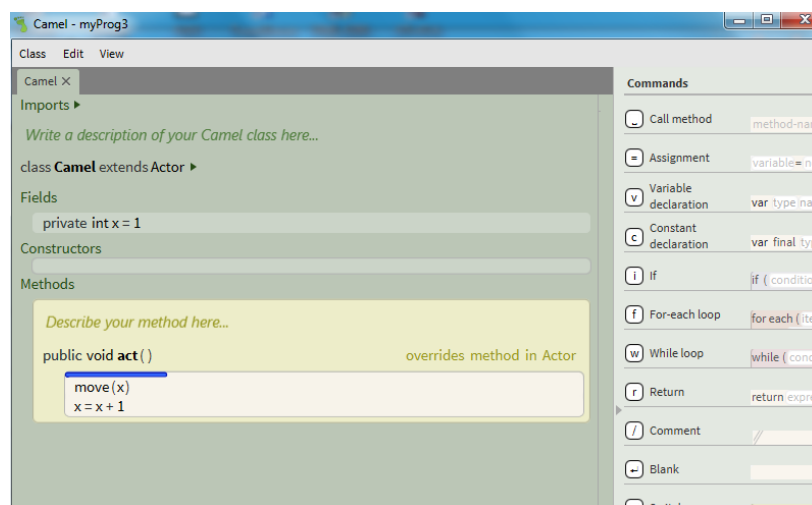


Figure 2.15: An Actor class using a Stride scenario (Greenfoot ver 3.0)

An analogy can be drawn between Java classes in object-oriented programming and objects of the APOS theory. A class in OOP, depicted in [Figure 2.15](#), is a virtual creation, but it becomes an object internal to computer memory, as internal as an object can become with Dubinsky's theory. The difference is that this class, which is a general description of an instance, is also created as mental objects in the learner's mind. This happens when the learner interacts with Greenfoot, engaging in mental constructions simulated within the “world” in Greenfoot through mental mechanisms. This is an internal mental construction of a design. Physical proof of this mental construction is found in compiling and executing Java code to enact the construction. The mental construction can be used and manipulated to create new schemas. The Greenfoot programming language now carries two ways in which a learner may explore Greenfoot – either a Stride or Java scenario. The idea behind Stride is to remove any unnecessary syntactical jargon from the file and allow a learner to directly devote more attention to his/her creativity instead of unravelling syntax.

[Figure 2.16](#) presents a typical Java scenario:

```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
/**
 * Write a description of class Camel here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Camel extends Actor
{
    /**
     * Act - do whatever the Camel wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
    }
}
```

Figure 2.16: Actor Camel class within a Java scenario (Greenfoot 3.0)

Unlike mathematics, these mental objects can be visualised with Greenfoot as a two-dimensional object having many properties, thereby removing the abstract/ concrete dichotomy criticised by Dubinsky (2000). Weintrop and Wilensky (2018b) coined Greenfoot as a frame-based editor as opposed to Scratch, which is classified as a block-based novice programming environment (NPE) (Zsakó & Szlávi, 2012; Papadakis et al., 2014; Papavlasopoulou, Giannkos & Jaccheri, 2019). Hazzan (2003) also sees understanding of abstraction as creating more connections to such an abstract concept. Greenfoot provides

that functionality to test any mental construction through mental mechanisms, where in mathematics, the answer or outcome of an applied mathematical concept may be correct according to the learner, but not correct according to the teacher or memorandum. A programming language such as Greenfoot can deliver immediate visual outcomes in terms of what was coded. The learner can obtain a definite result on his/her abstracted view, translated into Java code. The researcher regards this as an absolute epistemological view of the problem.

Dubinsky and McDonald (2001:2) uses the term “action conception” when an action is performed on physical objects or on objects (concepts) in the learner's mind. A genetic decomposition is when small portions of the schema are dissected and described in terms of the relationships that exist between schemas. When a learner is successful in his response to a problem and we can describe that response by means of a schema, we should know that the problem was *assimilated* into the schema, but if not, the existing schemas must be *accommodated* to handle the new phenomenon (Dubinsky, 1991). According to Dubinsky (2000), computer experiences can be used to help with reflective abstraction. Dubinsky (2000) refers to ISETL, a programming language used to illustrate APOS theory when looking at a function in mathematics. ISETL helped the learner increase his/her understanding of an Object and Process within APOS theory, using a computer environment, but confined to mathematics (Vidakovic, Dubinsky & Weller, 2018). The focus here is on using a programming language application and not on constructing an algorithm, which differs from computational thinking (Denning, 2017). As for this research, the focus is on constructing algorithms, using Greenfoot. This seems doable for mathematics-oriented learners, but it still places the learner within the mathematics realm, where mathematical anxiety is still a reality. The next section elaborates on the Greenfoot programming language as a computer experience to help with reflective abstraction.

(vii) The role of mental mechanisms within mental constructions in Greenfoot

Dubinsky (1991:106) indicates Piaget's 5 mechanisms needed for logical thinking, namely:

◆ Interiorisation

With a programming language such as Java, which consists of classes, the learner needs to understand the concept, but it is syntactically difficult to visualise and interiorise the outcome. Greenfoot allows learners to internalise by means of a visual interface creating these objects in a tangible way. The learner is able to involve his/her whole body within the Greenfoot programming language. The body of the learner will simulate the Actor object when developing algorithms to solve problems.

◆ **Coordination of actions in general**

This is when two or more classes interact in Greenfoot, e.g. *wombat* and *apple*, where the wombat eats the apple. The learner needs to consider both instances of these classes and provide code to make such objects interact and disappear. In general, the focus is on integrating several different and similar Actions to achieve a certain Process. It strengthens the relationship of the learner (subject) with the wombat and apple objects, which is abstraction per definition.

◆ **Encapsulation**

The code to perform certain steps can be enclosed within a method. This is when we apply encapsulation techniques, by hiding the specific code from the programmer within a 'capsule'. The method is then called via its name and messaging is used to communicate with internal structures or code. 'Refactor' is also a term used to make code tidy, not disturbing the essence of the intended code.

◆ **Generalisation of schemas**

This happens when an existing schema is applied to a wider range of phenomena and the learner becomes aware of other applications of such a schema. Sometimes certain pieces of code can be identified as being common to other classes as well, and may be moved into a class common to other sub-classes. As an example, specific animals, e.g., a dog and cat may belong to a common class called animal. All these animals may walk and run, which are common to specific animals. 'Animals' is a super class that may be inherited by all sub-classes like *dog* and *cat*. It takes a new mental construction within the learner to identify this Object from the existing Process and make it part of a schema called inheritance, among classes.

◆ **Reversal**

Reversal is the investigation of a process, backwards, by starting at the end and unpacking actions within a process that created a schema. These mental mechanisms are illustrated in [Figure 2.17](#) and can be applied in a programming language such as Greenfoot. The mental constructions take place and trigger thought Processes, thus strengthening the relationship of the subject with the Objects in confrontation. A schema, as identified by this research, must develop through construction and be subjected to continuous reconstruction. Reconstruction happens when a schema is subjected to Actions that lead to new processes and objects. A schema is therefore a dynamic entity, not a static entity (Dubinsky, 1991).

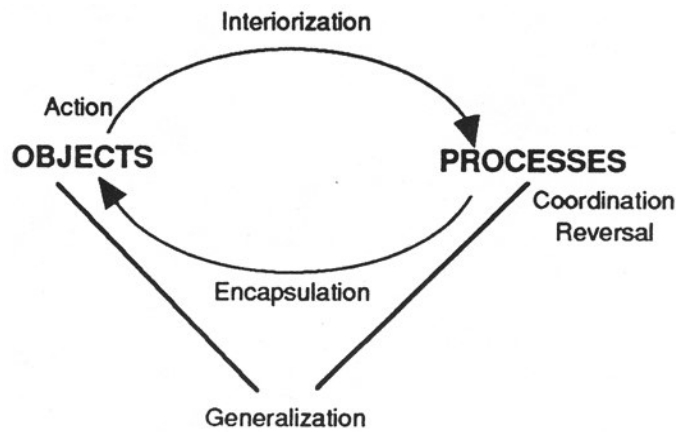


Figure 2.17: Schema and its construction (Adopted from Dubinsky, 1991:106)

According to Dubinsky (1991), schema construction stems from a genetic decomposition of schemas based on the five constructions of logical thinking. Dubinsky (1991) suggests a general structure for constructing a schema, but above all, the constructions and genetic decomposition of a schema allow the teacher to think of “how” and “what” needs to be taught, and for the learner, “how” to apply computational thinking.

Applying Dubinsky's APOS theory, the learner must implement the five constructs of logical thinking to be able to apply a GD of a schema construction, which forms the essence of intellectual thought. The advantages of programming are pointed out by Papert (1980, 2005), who shows that the learner may investigate aspects of science, mathematics, and the art of intellectual model building, when practicing programming. Papert (2005) also argues that programming creates experiences for learners to strengthen their intuitions and concepts on how thinking, learning and playing take place. Teachers need to understand APOS theory, but they will be confronted with terminology such as model, framework and formulation of a mathematical concept. According to Arnon et al. (2014:17), APOS focuses primarily on mathematical concepts. It is a model that points to the description of how to master mathematical concepts; a framework that explains how a learner mentally construct his/her understanding of such a mathematical concept using a genetic decomposition thereof; and is distinguished from a mathematical formulation of the concept, which looks at the positioning within the mathematical landscape. According to Piaget and Garcia (1989), learners use mental constructions, also perceived as stages within APOS theory, in order to understand a mathematical concept. In the next section, mental structures and mechanisms will be discussed to show how computational thinking skills may be promoted.

2.2.2.2 Research question (RQ) 2

RQ 2: How can computational thinking skills at a cognitive level of formal operations be promoted among high school learners through the teaching of a programming language aligned to Action Process Object Schema (APOS)?

To answer RQ 2, three SRQs were structured and analysed accordingly.

(a) Sub-research question (SRQ) 2.1

SRQ 2.1, “How are the constructs of a programming language taught among high school learners at a cognitive level of formal operations?” is answered in the next section by looking at (i) the meaning and importance of being “programming” literate, (ii) existing frameworks for programming language teaching, and (iii) a paradigm shift in teaching.

(i) The meaning and importance of being “programming” literate

It is important to discover how people learn as opposed to what their aptitude is. The focus needs to be on studying the process of “language learning” (White & Sivitanides, 2002:59). Hudak and Anderson (1990) perceive cognitive maturity and learning style as vehicles for academic success. White and Sivitanides (2002) focus specifically on identifying the hemispherical cognitive style and the stage of cognitive development necessary for different programming paradigms, as depicted in [Figure 2.11](#) (section 2.2.2.1(b)(iv)). The hemispheric dominance (cognitive style) and cognitive development are considered as human cognitive characteristics (White & Sivitanides, 2002).

Any mathematical activity, especially algebraic problems, stimulates the left brain, and so do certain programming language activities (White & Sivitanides, 2002). White and Sivitanides (2002, citing Riding, 1997) demonstrate by means of electroencephalograms (EEGs) that cognitive styles use different sides of the brain, thus indicating hemispheric differences. According to Gordon (1988), right and left cerebral hemispheres process information differently. Algebra and programming use the left side of the brain (Rotenberg & Arshavsky, 1997). The left hemisphere of the brain is used for probabilistic reasoning and the right hemisphere of the brain for deductive reasoning (Osherson, 1998). Another way of dealing with the issue is by using the Inventory of Piaget’s (1977) Development Tasks (IPDT) to measure a learner’s cognitive development for further research.

According to White and Sivitanides (2002), the age group from 11 to 12 years of age onwards is when younger learners acquire the skill of formal operational thinking (Piaget, 1964; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017). The transition from concrete to logical/ abstract thinking may occur much later, or not at all (Griffiths 1973; Schwebel 1975; Pallrand 1979; Young, 2012).

According to White and Sivitanides (2002, citing Losh, 1984), Fletcher (1984), Little (1984), Ott (1989) and Monfort, Martin and Frederickson (1990), cognitive development (what can be learnt), cognitive styles (how learning takes place), and prior experiences are aspects to consider when learning OOP. Cegielski and Hall (2006) also regard personality as an

important enabler for learning OOP. Barr, Harrison and Conery (2011) argue that computational thinking is much more than just cognitive ability. Prior experiences may be important for shaping personality, but personality, according to Cegielski and Hall (2006), revolves around self-esteem, generalised self-efficacy, locus of control and neuroticism. This shows that personality is so much more than mere experiences.

According to Wright, Rich and Leatham (2012:4), the Computer Science Teachers Association (CSTA) provides four motivational factors for why programming literacy is important, namely: (i) programming links to other fields; (ii) it teaches problem solving; (iii) it engages every learner; and (iv) it provides greater employability. These points are extremely broad and the researcher would rather consider the motivational factors of Szlávi and Zsakó (2017).

Wright, Rich and Leatham (2012:5) argue that being literate, whether reading literate or programming literate, means to “manipulate the object of literacy, e.g. communication technologies, in a meaningful way”. They draw an analogy that being able to read, does not imply being literate. Prensky (2008) added programming as a new form of literacy expanding the current 3Rs to 4Rs.

(ii) Frameworks for teaching PLs

To become programming literate might depend on the framework being used. One such framework is the **Pedagogical Content Knowledge** (PCK) framework (Saali et al., 2011) in [Figure 2.18](#). This PCK framework originated from Shulman (1986) who suggested, “The ways of representing and formulating the subject that make it comprehensible to others” (Shulman, 1986:9). The domain of Information Technology using personal computers was still in its infant stage at the time and the framework described all content under one umbrella. Authors such Cooper, Pérez and Rainey (2010) and Brennan and Resnick (2012) also proposed and implemented frameworks around computational thinking from specifically an IT perspective using Scratch. Papert (1980), on the other hand, advocated teaching strategies to accomplish the learning of PLs. Many authors agree that a PCK only develops with years of experience (Grossman & Lynn, 1990; Rovegno, 1992; Sanders, Borko & Lockard, 1993; Van Driel, Verloop & De Vos, 1998; Morine-Deshimer & Kent, 1999; Loughran et al., 2001).

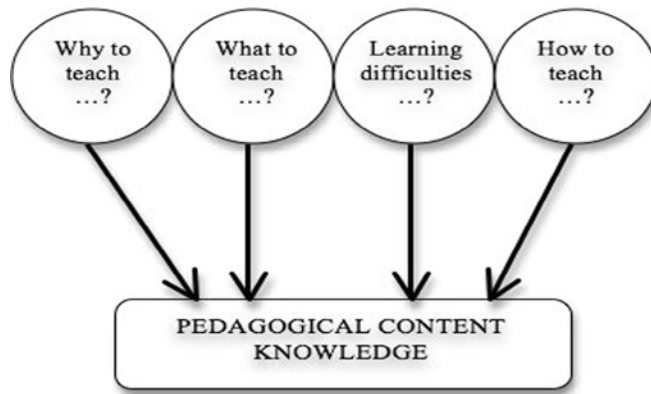


Figure 2.18: Diagram based on Grossman's Reformulation of PCK (Adopted from Saeli et al., 2011:76)

Grossman's diagram (adopted from Saeli et al., 2011:76) (Figure 2.18) highlights **what** should be taught, **who** needs the teaching and **how** the education will be rolled out. Although Saeli et al. (2011) used the basic PCK Shulman's (1986) PCK was already augmented by Koehler and Mishra (2009) to include technology specifically. The technology domain is complex and should be a separate grouping within the Shulman Framework. Koehler and Mishra (2009) restructured the PCK to a Technical, Pedagogical and Content Knowledge (TPACK) framework, depicted in Figure 2.19.

The Koehler and Mishra framework (2009) focuses on (i) a content domain, i.e., knowledge about WHAT must be taught, (ii) a pedagogical domain, pointing to teaching and learning about HOW teaching will be rolled out, and (iii) a technical domain, pointing to the platform and resources necessary for a successful rollout. The remote desktop protocol technology (RDP) on a Windows server was used by the learners during this research. The rollout of Virtual Machines was a project in progress, which may have increased the speed of delivery. The RDP makes use of shared memory and the more desktops joined, the slower the execution of a scenario becomes. This technical knowledge influences the rollout of the content and pedagogical approach. The cognitive load theory heavily relies on such a framework as TPACK to ensure a decrease in cognitive load by limiting an extrinsic load through proper content delivery and the way information is represented. The more complex learning becomes, the less learners rely on support and guidance, which shows a decrease in extraneous load, until new complex learning demands an increase in extraneous load again (Sweller et al., 2019).

The following TPACK framework may be used to describe implementation of technology in education and also for this research to promote cognitive load theory:

- What programming language to be taught to high school learners as well as the hardware involved. Technical knowledge (TK) about the application of the programming language and platforms to best achieve the outcomes of the approach
- Programming language syntax and possible algorithms and constructs of the programming language to facilitate computational thinking; the storyboard techniques for the LMS are used to compliment training Content knowledge (CK) about the subject matter being taught involves multiple disciplines
- What are the best practices to teach and use for a programming language, computational thinking and LMS? Pedagogical knowledge (PK) about each specialised field, namely, programming language, computational thinking and LMS need thorough research and understanding

The TPACK framework provides a shift from WHAT to HOW. These domains cannot live in isolation and need to be integrated, as illustrated in [Figure 2.19](#). These three domains are interconnecting as Technical and Content Knowledge (TCK), Technical and Pedagogic Knowledge (TPK), Pedagogical and Content Knowledge (PCK) and the intersection of Technical, Pedagogical Content Knowledge (TPACK). Overall, the augmented TPACK is just a framework and needs a specific conceptual strategy from a research perspective.

The original PCK framework focuses on “why the high school learner should learn to program” as opposed to “why learners should be taught programming at all”. The complexity of adding a visual programming environment into the mix makes manipulation of IT-related content implicit. TPACK (Koehler & Mishra, 2009) would thus provide a better focus on IT, and specifically, a programming mix within the PCK mix. The focus is on how computational thinking is promoted when using programming concepts and APOS theory as lens, with a constructionist approach.

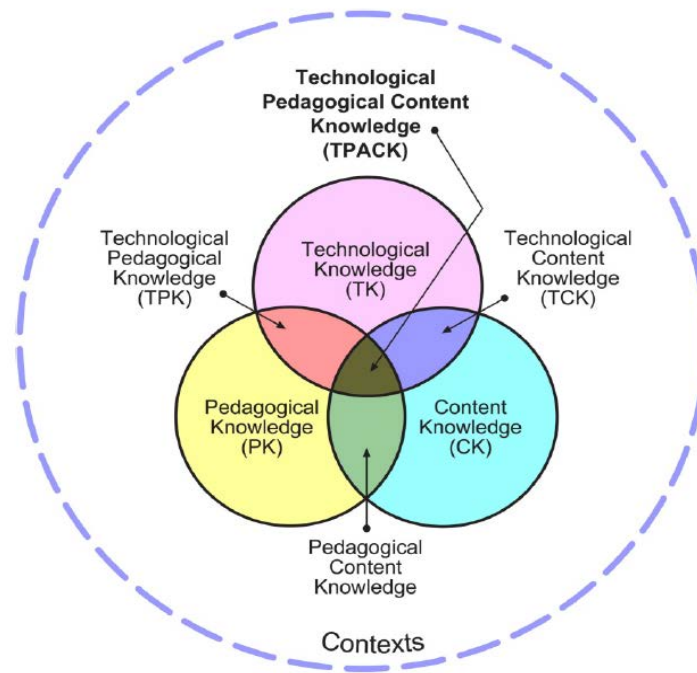


Figure 2.19: TPACK framework accommodating technology (Adopted from Koehler & Mishra, 2009:63)

Highlighted by Cegielski and Hall (2006), a person exhibiting the skill of being fluent in OOP is a computational thinker who has or is acquiring a cognitive level of formal operations. Aspects like the relationship of values, cognitive ability and personality cannot be ignored within any framework. Such a value component is called theoretical value belief.

The TPACK of programming (Koehler & Mishra, 2009), namely “why teach a programming language?”, “what to teach?”, “what are the problems in learning a programming language?” and then deciding on “how a programming language should be taught” may focus on the structure/syntax of the programming language. Syntax and structure may blur the importance of solving problems using the programming language. Developing algorithms to solve problems are necessary to engage learners in using APOS theory within the programming language to achieve computational thinking. Leaving the problem solving component for the next level may remove the very essence of the advantage in teaching a programming language in the first place. Complexity of syntax should be minimised or even removed altogether, so that the learner only focuses on problem solving, gathering implicit syntactical knowledge, also known as tacit knowledge. Considering [Figure 2.11](#), a programming language must provide a progression path, moving through the levels of cognitive development and allowing the learner to grow through the usage of computational thinking (White & Sivitanides, 2002). Weintrop and Wilensky (2018b) support this progression path, but position the experience within modality programming environments and the learning environment. Modality refers to the representational infrastructure used and the set of

interactions supported by the interface (Weintrop & Wilensky, 2018b). Szabo et al. (2019) sees modality as the method by which the user creates the program, as visual blocks or text. The secret is to blend block-based programming with text-based programming. Greenfoot programming language can be classified as a dual-modality programming environment that supports both block-based and text-based programming. The concept of supporting novice programming environments (NPEs), as discussed by Papadakis et al. (2014), is not the focus of this research, as this research focuses on the cognitive level of formal operations and thus supports a dual-modality programming language.

(iii) Paradigm shift in teaching

Considering all the discussions above regarding computational thinking, abstraction, logical thinking and the like, this research takes on the interpretation of Dubinsky's (1991) APOS theory. This research employs APOS theory from Piaget's reflective abstraction as a vehicle for constructing programs using a programming language with a dual-modality programming environment. This is the challenge in SA which needs to be addressed, in that learners must reach that level of thinking where Dubinsky's research starts, hence focusing on learners aged 11 to 15 years. Once again, the choice of age is based on a development stage called formal operations, as identified by Piaget (1964, 1975, 1977), during his studies. This research relies on more than 40 years of cognitive development research by Piaget. Dubinsky (1991) further points out that his interpretation of Piaget's work describes the epistemology of mathematical concepts, which should explain the challenges learners may have when mastering these concepts, and hence may influence the design of instruction (when administering these mathematical concepts).

One basic reason for not using mathematics to understand mathematics is the abstract nature of mathematics and the various belief systems about mathematics that exist (Moscucci, 2007). It can be a challenge explaining an abstract concept to someone using the same abstract concept with no concrete entry points. The meta-belief system activity (MBSA) of Moscucci (2007) was not used, but the semi-structured interviews and questionnaires (Appendix B), answered by learners in this research show that mathematics teachings are challenging.

The reason for attaching Dubinsky's research is the epistemological nature of his research being acceptable when criticising all researchers in the field of computational thinking. Mathematics is abstract in nature and so is a programming language (PL), but in using a programming language like Greenfoot, its interiorisation capabilities are much more advanced than the normal PLs such as pure Java, C# and the like, and above all, Greenfoot enjoys a greater degree of interiorisation than mathematics does. Tall (2008) sees

embodiment as a prominent factor in moving up and down the ladder of conceptual embodiment to proceptual symbolic until the learner is susceptible for the axiomatic-formal world. Greenfoot is built around embodiment as illustrated during the interventions in this research.

(b) Sub-research question (SRQ) 2.2

SRQ 2.2, “How do the constructs of a programming language align to APOS among high school learners at a cognitive level of formal operations?” is discussed in the next section by (i) how learners are taught programming in a less painful manner, (ii) reflective abstraction in Greenfoot, (iii) mental structures and mental mechanisms, (iv) when should learners engage in learning a programming language, and (v) using an example to show how mental structures are applied.

(i) How can learners be taught programming in a less painful manner?

The question is, “How to teach programming then?” Saeli et al. (2011) quote Hromkovič (2006) who states that programming must be seen as a skill to communicate in an unambiguous manner. To minimise all the challenges stated earlier, the programming language should be simple enough, yet encompassing the goal of instruction and focusing mainly on semantics. This may lead to fewer syntax errors because of the nature of current tools having interactive design qualities, as Brennan (2012) as well as Weintrop and Wilensky (2018b) point out when using Scratch as an NPE (Papadakis et al., 2014). Syntactically, Linn and Dalbey (1989) regard program comprehension and program generation as a chain consisting of three links, namely, single language features, design skills, and problem-solving skills. Du Boulay, O’Shea and Monk (1989) see the need of a framework or model, to have a guideline when involving learners in programming which should still be applicable in dual-modality programming languages. The model should aim at their age, background or type of studies. Currently, interactive media or block-based visual programming is available, and hence Brennan (2012) sees constructionism as an important vehicle with the emphasis on design when young learners use interactive media like Scratch. Greenfoot, on the other hand, is also a constructionist vehicle and allows for embodiment from the learner’s perspective.

Saeli et al. (2011) argue that languages like LOGO (Feurzeig et al., 1970; Papert, 1980; Resnick & Ocko, 1990), Scratch (Resnick et al., 2009), Alice, Greenfoot and Gamemaker (Cooper, Dann & Pausch, 2003; Kölling & Henriksen, 2005; Overmars, 2005), to mention a few, are based on Piaget’s (1997) model of learning. It allows learners to focus on semantics, removing complexities from learning a programming language. Having used and taught LOGO, it is similar to Greenfoot as a dual-modality programming language, except that LOGO is a procedural language as opposed to Greenfoot being an OOP language.

(ii) Reflective abstraction within Greenfoot

The cognitive level of formal operations as researched and coined by Piaget (1985) (Dubinsky, 1991) is a sufficient development stage in the life of any person for him/her to be able to build mathematical structures using concepts such as commutativity, number, trajectory, see-saw, multiplication and fluid levels. These mathematical concepts are formed during the development stages of a child and should be present within the stage of formal operations. Within Piaget's (1977) research, Dubinsky (1991) found various forms of mental mechanisms within reflective abstraction necessary for mathematical thinking. By using reflective abstraction, the mental mechanisms are:

- The ability to interiorise, i.e. making sense of pictures, mental images, symbols, language, etc.
- “General coordinations of actions” in using actions to construct new ones (Dubinsky, 1991:97)
- Encapsulation of a Process (dynamic) into an Object (static). Dubinsky (1991) refers to Piaget's (1985) statement as when “... actions or operations become thematised objects of thought or assimilation” (Piaget, 1985:49, cited in Dubinsky, 1991:100). The author perceives this as the construction of structures
- Generalisation of schemas, when an existing schema is applied to a wider range of phenomena and the learner becomes aware of other applications. Typically, this can be seen when learners are taught some aspects, but they must be able to see the value of the teaching in a broader framework of applicability
- Think of an internal process in its reverse form

(iii) Mental structures and mechanisms

In order to comprehend mental mechanisms necessary to construct mental structures, this research uses APOS theory as a vehicle to do just that. The researchers can provide a conception of the APOS theory concept as put forward by Arnon et al. (2014), that the schema is an encompassing term, which houses Actions, Processes, Objects and maybe other schemas. These schemas describe the learner's mathematical conception of a mathematical concept, how it is put together organisationally, and an operational roadmap of the mental structures that it consists of. According to Arnon et al. (2014), APOS theory can be used as both model and framework. The former describes *how* to master mathematical concepts and the latter explains how individuals go about mentally to understand these mathematical concepts from a cognitive perspective. Dubinsky (1991) puts Piaget's (1977) reflective abstraction to work in mathematics and sees reflective abstraction as when mental objects are constructed and mental actions on those objects are executed through mental mechanisms. The author further argues that he incorporates a schema (collection of objects and processes) which will be invoked by a learner in order to grasp a problem situation, but

based on prior knowledge or some specific concept of the domain (mathematics in this case) in question.

This research reflects on the role of reflective abstraction in a frame-based editor (Weintrop & Wilensky, 2018b) and the semi-visual programming language, Greenfoot, in order to create an analogy whereby computer programming language objects and actions on those objects transformed into mental objects and actions on those objects. Using a semi-visual programming language like Greenfoot to initiate programming is a rather foreign mental construction. However, it should aid in the assimilation of schemas, but even more in the accommodation of schemas by changing those “linear” or preformed mental structures, to allow quicker assimilation of schemas and synthesis as learning progresses.

(iv) When should learners engage in learning a programming language?

According to the Neo-Piagetian stage of cognitive development, primary school learners may not be so far up the ladder of cognitive development (Young, 2012:242) as depicted in [Figure 1.2](#), as opposed to high school learners, and the methodology needs to be changed in order to reach the same goals in problem solving. The researcher’s exposure to the South African educational system as a registered teacher at primary, secondary and tertiary levels has led him to believe that the focus of the DBE in particular should start at a much lower level than Grade 10 and not a selected group only.

The actual usage of technology by teachers should empower them to make Information Technology available to younger learners from as early as Grade 6, when the necessary mathematical emphasis on principles are taught. Teachers may argue that mathematics is doing just that, but learners grapple with mathematics because of the subject being abstract and directly addressing the level of formal operations. The DBE and other educational departments should rather offer IT for more than just the elite learners who eventually are not using the man hours and money spent by the tax payer and government to educate them in skills they will not pursue enrolling for courses such as IT at a university.

Research conducted by Kranch (2010) shows that acquiring a skill such as programming, takes a long period of time. This statement depends on what is meant by the term programming. If programming refers to block-based programming within a novice programming environment (NPE), the entry level is not as severe as text-based syntactical programming languages. As pointed out earlier, many authors regard qualities such as computational thinking and cognitive ability not as independent entities, but as entities resting very much on human characteristics of personality and theoretical value beliefs (Cegielski & Hall, 2006; Wing, 2006; Barr, Harrison & Conery, 2011).

(v) Applying mental structures as an example

Before having an in depth look at the mental structures, the APOS acronym needs to be broken down into manageable parts and written down or typed out, i.e.:

A – Action

P – Process

O – Object

S – Schema

The reader needs to act and make sense of the acronym, also known as the interiorisation mechanism. These actions may be done by means of writing down the acronym and memorising each word associated with the letter. It is easily memorised in that “acronym” is a word pronounced formed from the initial letters of words (Pearsall & Trumble, 2002). According to Dubinsky (1991), the **A**ctions taken are all external and act as a stimulus coming from the outside of the reader.

Once the reader can visualise the acronym in his/her mind without writing it down, it is interiorised within the reader’s mind, also known as a **P**rocess. APOS theory is an abstraction, but the current state is a fuzzy abstraction to the reader. Currently, the APOS theory concept and the conception of the reader may differ, more than likely. Although the teacher or reader may encapsulate the process into an **O**bject that represents the mental constructions for an individual to enable an understanding of APOS theory as a model or a framework, the abstraction process needs further analysis to remove the fuzziness. The reader sees APOS as a totality in that the reader encapsulates the process into a cognitive object, which may have meaning as either a model or framework to the reader (Dubinsky & McDonald, 2001). The reader has now taken **A**ction to create a **P**rocess of the APOS acronym to form an **O**bject with the role of a model or framework within the mind of the teacher or reader.

The last letter “S” constitutes the word “Schema”, which points to the long- and short-term memory of the teacher or reader. Several schemas pre-exist within the reader or teacher’s mind. The fact that the reader is reading this thesis shows that the reader does have some interest in constructionist learning or in the constructionist approach during teaching and learning, and that some schema(s) do exist in his/her mind. The challenge is firstly to understand what constitutes a schema, and secondly, how does one expand an existing schema(s)? To answer this, one needs to compare the APOS theory concept to the reader’s conception through a reflective activity (Arnon et al., 2014). One can perform a genetic decomposition to understand how APOS theory’s mental constructions are related to form a

larger mental structure called “Schema” (Arnon et al., 2014). According to the authors, genetic decomposition can be inferred from data generated by analysis of observations or interviews and from the historical development of a concept. The genetic decomposition acts as a mechanism, which enables teachers or researchers to understand whether the learner mentally understands a concept, or what the difficulties are. In using such a genetic decomposition framework, the analysis of data is looked at from a similar perspective, which gives more reliability to interpretations from different researchers. Furthermore, a genetic decomposition is about how the actions are interiorised into **Processes**, and how the **Processes** are encapsulated or coordinated into **Objects**. Care must be given to what these **Actions**, **Processes** and **Objects** exactly are; it must not merely become confirmation of taking action, resulting into a process that is encapsulated into an object, without stating exactly or discovering the actual detailed actions, processes and objects involved in the mental construction of a conception when understanding a concept. The focus should thus be on the mental constructions of the learners when a conception is formed from a concept. Within the genetic decomposition, a **Schema**, or certain schemas, will be prerequisites to understanding a concept (Arnon et al., 2014).

The relationship between solving a mathematical problem based on a mathematical concept and writing a program using a computer programming language to map an algorithm, is different in that a multitude of programming concepts are used simultaneously for the latter. Connolly, Murphy and Moore (2009) argue that computing is abstract in nature and that the problem-solving activity generates anxiety among learners. The cognitive emotional and physiological states that cause anxiety need to be addressed to improve confidence and competence, similar to mathematics learners. This research is an investigation into the conceptualisation of concepts in object-oriented programming languages using APOS theory for mathematical concepts. Five basic concepts exist in any programming language, namely, variables, control structures, data structures, syntax, and tools (Brennan & Resnick, 2012). The researcher uses the Greenfoot gaming programming language that entails a broader category as opposed to normal programming language concepts, namely OOP. Concepts within OOP include dynamic lookup, abstraction, subtyping and inheritance, supported in Greenfoot as a derivative of Java. Abstraction and inheritance together with concepts identified by Brennan and Resnick (2012), which forms part of the Greenfoot programming language, were investigated in this research.

[Figure 2.20](#) depicts the research on discovering a programming concept as a hypothetical mental process.

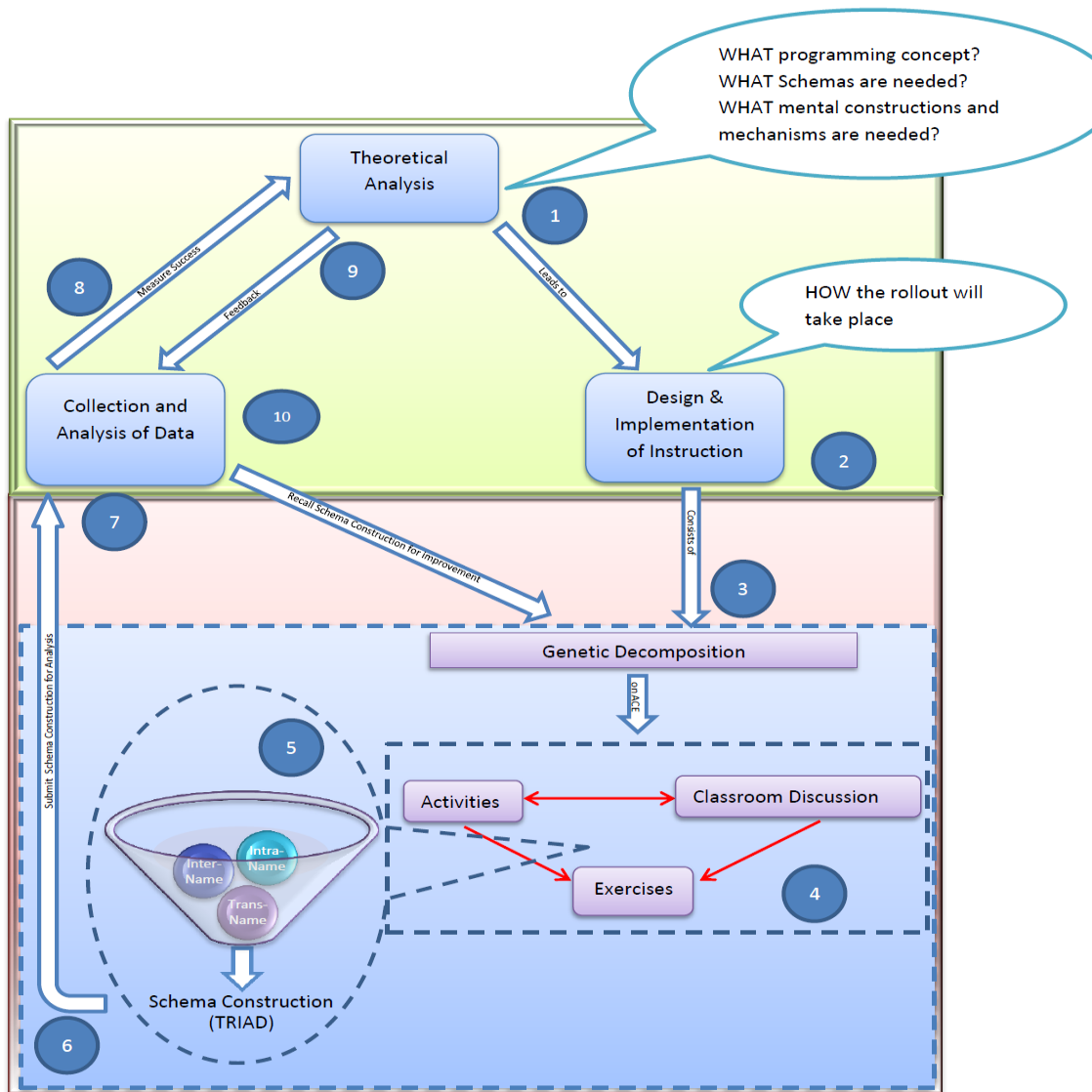


Figure 2.20: Interpretation and adaptation of APOS theory from a research and curriculum development stance (Adapted from Arnon et al., 2014:112)

A brief discussion of Figure 2.20 follows next.

Step 1: Theoretical analysis

A theoretical analysis is required to determine what concepts and thus schemas are in question or needed to learn a programming concept. According to Arnon et al. (2014), the theoretical analysis is thus a driving force for the design and implementation of instruction in those mental constructions identified in the literature analysis of this research. It can thus act as a barometer to measure if learners indeed committed to the identified mental constructions and how well the learners learnt a programming concept.

Take a simple concept in the Greenfoot programming language. Learners may attend your class for the first time, where the instructor needs to introduce concepts such as:

- Tools necessary to run a Greenfoot scenario

- Structure and design of a Greenfoot scenario
- Successful rollout of the implementation of a Greenfoot scenario within the world of a developer

Mental structures to consider within APOS theory are **A**ctions, **P**rocesses and **O**bjects to eventually make up a **S**chema in order to find a place/niche for all Greenfoot scenarios. At this stage, learners come with an existing Schema or none other PLs. An editor and compiler are used to create object code and some degree of debugging may suffice to create and execute code using the Greenfoot programming language. The aim is to identify mental mechanisms that will help with the design and implementation of instruction when running any Greenfoot scenario, once constructed, which is different from the existing schema. For this example, the mental mechanisms may be interiorisation, coordination and encapsulation, where the **O**bject of instruction is to interiorise all those **A**ctions into a **P**rocess called “running of a Greenfoot scenario” and that may be encapsulated into an **O**bject called “Greenfoot Application”. Hopefully, the **S**chema of how to deal with the Greenfoot editor, structuring a Greenfoot scenario, and then running (executing) that scenario may be regarded as two **S**chemas. The structure of a Greenfoot scenario **S**chema and the usage of the tools **S**chema must be coordinated and assimilated into the existing **S**chema of the testing and debugging of a computer program. Here it is not only about the tools, but also about the structure of the Greenfoot scenario. The two **S**chemas can be investigated separately, but coordinated into one final **S**chema called “running of a Greenfoot scenario”.

Step 2: Design and implementation of instruction is needed to pave the way forward

According to Arnon et al. (2014), the design and implementation of instruction starts off with a preliminary genetic decomposition (GD). One needs to anticipate how the mental constructions may take place and it needs to be tested against step 1. It is also considered as preliminary, for one must still test this against the theoretical analysis as described by Arnon et al. (2014).

Step 3: A Genetic Decomposition (GD) is used to give meaning to activities, classroom discussions and exercises (ACE)

For genetic decomposition, mental structures and mental mechanisms (see section 2.2.2.1 (c)(vii)) give prominence to ACE (Figure 2.20). Mental structures, which need to be acquired through mental mechanisms, must then be combined into the stages of building a schema, also known as the TRIAD, where intra-, inter- and trans-stages are deployed in alignment with the specific schema in question.

Step 4: ACE re-enforces the classroom discussions based on the activities

The activities, classroom discussions and exercises identified, make the genetic decomposition (GD) a reality, according to Cottrill et al. (1996). The authors used the following [Table 2.1](#) to state the structures and mechanisms identified in step 1. The actions and processes of the GD are refined by describing how the learner will accomplish the **A**ctions and **P**rocesses.

Table 2.1: Genetic Decomposition (Adapted from Arnon, 2014:27)

Preliminary Genetic Decomposition	Refinement
1P: The Action: Viewing videos illustrating the tools necessary for programming a Greenfoot scenario	1.1R: The learner logs into my Moodle site and familiarise him/herself with such an LMS. It is just a matter of clicking on the hyperlinks to activate the videos. 1.2R: The learner forms part of an activity, also known as a flipped classroom technique, where he/she watches a video on the implementation of these tools. 1.3R: The learner takes action by opening Greenfoot application interface.
2P: The Process of Interlortizing the Action to activate Greenfoot and be cognicance of running as an RDP sessions or stand alone application.	2.1R: Create a new scenario in Java and create a World and an Actor that will live in that world.
3.1P: The Action of opening Greenfoot scenario, create any World and Actor and save the appropriate folder where Greenfoot scenario will be serviced by Greenfoot.	3.1R: The learner adds a World subclass and an Actor sub class and compiles the schenario.
3.2P: The Action watching the video on a basic Greenfoot soenario within the context of creatig a game.	3.2R: The learner mimics the video and type out the structure of the Greenfoot soenario.
3.3P: The Process is created by Interlortizing the steps of such a Greenfoot scenario.	3.3R: Now create a Greenfoot scenario without looking at those steps from some external medum, such as video or textbook. Save the script at the correct location

Step 5: Schema construction

Each component of ACE is examined by means of APO (actions, processes and objects). APO gives rise to a schema. The schema consists of intra-, inter- and trans-stages. The last part in APOS theory refers to a schema. Arnon et al. (2014) describe a schema as a study conducted on the relations and constructions which are built during problem solving. This study should show the structure of the learner's schema, which is also unique to that person. Arnon et al. (2014:112) cites Piaget and Garcia (1989) using the "triad" that consists of three stages to describe the progression and development of such a schema. Trigueros (2005) views schema development as an effective way to understand the construction of schemas.

Any schema, according to Piaget and Garcia (1989), develops through intra-, inter- and trans-stages linked to that schema's name.

A real life mathematical example to relate is discussed next:

When viewing graph mapping towards a linear equation of a straight line, the **intra-linear stage** focuses on the plotting of coordinates and represents these coordinates on graph paper. It is assumed that the learner does possess a schema on positive and negative numbers and x and y axes. The actions taken by the learners in plotting the coordinates force them to observe in which quadrant an x and y coordinate with a specific sign will reside, and after a while the learners will interiorise the quadrants of $(-x,y)$, $(x,-y)$, $(-x, -y)$. The **inter-linear stage** commences when the learner develops more knowledge on coordinates and infers the sign of an x or y coordinate which relates to a specific quadrant on the graph.

Step 6: Action reverts back to collection and analysis of data

If the genetic decomposition (GD) was successful, the GD will be used to strengthen the mathematical concept for learners. If not, the GD will be reconsidered using theoretical analysis. From this step onwards the GD may be successful using formative assessment or the GD will be taken back to the drawing board again to design and implement instruction to repeat the cycle. Cognitive load theory remains a factor in the mix and an LMS is a proposed way added to the extraneous load when complex learning is involved. The inclusion of an LMS is discussed in the next section.

(c) Sub-research question (SRQ) 2.3

SRQ 2.3: How does the use of an LMS, as a platform for learning, aid the teaching of a programming language aligned to APOS to promote computational thinking skills at a cognitive level of formal operations among high school learners?

Baist and Pamungkas (2017) are concerned that learners do not receive sufficient programming examples, and the lectures may be of poor quality in general. Their results show that learners need more activities and discussions to acquire programming skills. The access to an LMS that is properly structured by an instructional designer may assist in reducing the concerns expressed by the authors who maintain cognitive load theory.

Prensky (2008) argues that programming must be regarded as a new form of literacy (section 2.2.2.1(c)(iv)). The process of advancing any learner to acquiring the skills of the fourth "R" is a Piagetian challenge. According to White (2003), other prerequisites such as a different skills level required for mathematics, with specific mention of algebra, are prerequisites for the fourth "R". Wing (2006) however sees computational thinking as the

fourth component, not programming. One way in which to uphold cognitive load theory is to allow access to an LMS in order to balance learning in a constructionist way. Ways to optimise cognitive load through an LMS is discussed next in (i) perspective on an LMS, (ii) novice versus expert programmer, and (iii) the worked example effect in the next section.

(i) Perspective on an LMS

According to Schober and Keller (2012), researching the usage of an LMS to assist learners with blended learning did not have the desired effect. Learners have not used the LMS as expected, but the authors also state that the LMS was rolled out not using a proper instructional design approach. The interface was frustrating to learners, which may be the cause of the negative outcomes. In such an event, the basic principles of upholding cognitive load theory are violated by increasing the cognitive load extraneously as a result of an incorrect instructional delivery method (Mostyn, 2012; Sweller et al., 2019). The LMS used for this research is the Modular Object-Oriented Dynamic Learning Environment (Moodle), which is open source technology selected for this research with the purpose to save costs with the implementation at schools. Alomari et al. (2020) state that human factors contributing to the usability of the LMS need to be considered, which may be the answer to the findings of Schober and Keller (2012), namely that learners did not use an LMS as expected. The focus should be on LMS effectiveness, which can be measured in terms of the duration it takes a learner to perform a task using an LMS, thereby promoting user satisfaction (Alomari et al., 2020). User satisfaction points to the belief of users that the information system to their disposal meets their information requirements (Eom, 2014).

(ii) Novice versus expert programmer

The speed at which a novice can become an expert in programming is dependent on worked examples to speed up learning (Abdul-Rahman & Du Boulay, 2014). Van Gog, Paas and van Merriënboer (2005) regard programming language expertise creation as optimising learner working memory (LWM) by packing more information into fewer elements, hence releasing more memory that can be used for problem solving. The cognitive load should become less for the learner (Sweller et al., 2019). The short-term working memory of a human being can only handle four independent chunks (Cowan, 2001), but is influenced by long term memory activity. Becoming a programming language expert is a process of “internalising, organising and automatising domain knowledge and skills” (Kranich, 2010:76). The process consists of three stages. The first stage is where knowledge held by the novice is poorly organised. In the second stage, the knowledge becomes hierarchical and understood. The third stage refers to knowledge reorganisation, containing associations across knowledge levels to create a personal complex knowledge repository (Kranich, 2010). This theory coincides with

Piaget's (1977) two stages of thinking, embedded in organisation and adaptation (Woolfolk et al., 2003).

To become an expert in a pure syntax-driven programming language, Zeitz and Spoehr (1989:327) state the following stages:

- The novice should be taught the basic syntax of the language, only focusing on the basic features of the language, called a breadth-first hierarchical organisational approach
- After understanding the syntax, program plans and semantics (logic rules), i.e., an experiential or problem-based instruction should be followed, as knowledge becomes “hierarchical, orderly, and easily verbalised”
- Thirdly, the plans and problems being explained should increase in complexity as expertise grows, and the instruction based on problem solving should be the chosen method used. The knowledge remains hierarchical or scaffolded, being available at “various levels of abstraction with important associations across and within levels of abstraction”

The arguments of Zeitz and Spoehr (1989) may describe the final outcome of the expert, but research brought visual programming languages into being. Visual programming languages allow a gradual growth in complexity, thus making programming more accessible to everyone, but with the aid of worked examples found on an LMS. Cognitive load theory has an impact on the views of Zeitz and Spoehr, although their stages are still valid, where the abstraction levels take on a different almost easier approach in visual systems. Having advocated that instruction should be based on a chosen method of problem solving, the “worked-example effect” is discussed in the next section.

(iii) The worked-example effect

As highlighted earlier, the LMS may provide a focused area of interest or knowledge domain that the learner may access to complete a task. A programming language is a new concept and an LMS may alleviate that uncertainty and frustration of where to research such a task handed out to learners. The LMS thus provides domain-focused worked examples in programming that support cognitive load theory (Clark, Kirschner & Sweller, 2012; Li, 2016; McPhail, 2016; Sweller et al., 2019). The primary objective of cognitive load theory is “the generation of novel instructional techniques” (Sweller & Paas, 2017:86). According to Li (2016:58), many effects are developed from cognitive load theory, such as the “goal-free effect, worked example effect, completion problem effect, split-attention effect, modality effects, redundancy effect, and variability effect”.

2.2.3 Theoretical conceptual framework

A framework must “possess ontological, epistemological and methodological assumptions, where each concept within the conceptual framework plays an ontological or epistemological role” (Jabareen, 2009:51). Conceptual frameworks point to a network of interlinked concepts. The ontological assumptions or the ‘way things are’ are supported by Bachelard (1938) and Brousseau (1983) in context of the theoretical conceptual framework, depicted in [Figure 2.21](#). The theoretical conceptual framework also depicts epistemological fraud (D’Amore, 2008; Jankvist & Niss, 2018). The researcher resolved the epistemological fraud through dual-modality computational notation called Greenfoot. The framework was developed through a qualitative analysis of the literature review. The proposed conceptual framework is based on data collected during Educational Design Research (EDR) using literature reviews, interviews and practices (Jabareen, 2009) as indicated in Chapter 5. The EDR process was iterative and led to a few intermediate theoretical conceptual frameworks.

The concepts investigated, as depicted in [Figure 2.21](#), are literacy skills, cognitive ability, cognitive loads, value beliefs, critical thinking, mental structures, computational thinking, didactical situations, abstraction, complexity cloud, programming language, learner management system, mathematics and science.

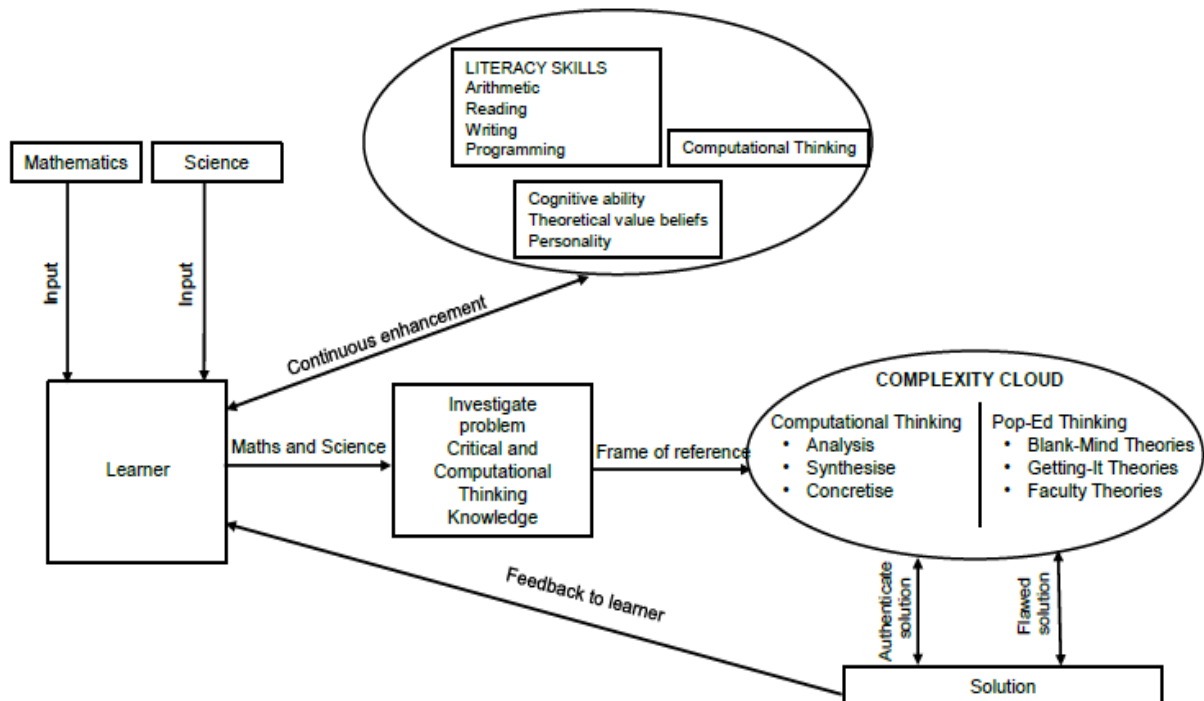


Figure 2.21: The theoretical conceptual framework for the improvement of computational thinking among learners

In Figure 2.21, the learner receives input through learning computational thinking subjects such as Mathematics and Science. The schemas of such a learner are updated in terms of the schemas of concept definitions within Mathematics and Science. As depicted in [Figure 2.21](#), the cognitive environment of the learner is influenced by literacy skills, which should include computational thinking, influenced by cognitive ability, theoretical and beliefs about mathematics and the learner's personality (Cegielski & Hall, 2006; Moscucci, 2007; Prensky, 2008; Moscucci & Bibbo, 2015). The learner receives lessons and tuition in class and is expected to investigate concepts, problems through critical thinking and computational thinking. In order to accomplish such a task, the learner finds him or herself within a complexity cloud of computational thinking and Pop-Ed thinking (Papert, 2005) that influence the learner's academic functionality.

The complexity cloud depicted in the theoretical conceptual framework is based on research conducted by Bachelard (1938), who argued that existing cognitive beliefs might affect the progress of science. Papert (2005) further argued that thinking processes and beliefs become obstacles to understanding mathematical concepts as researched by Flavell (1976), Moscucci (2007) and Jankvist and Niss (2018). McGowen and Tall (2010:170) posited that learning can be impeded by "met-befores". Brousseau (1983) did not use the term "met-befores", but posited that certain existing knowledge may prevent the acquisition of new knowledge, also known as an epistemological obstacle. This also holds true for the didactical situation (Brousseau, 2002) in the class at the private school and in general where teachers are unaware or not concerned about the beliefs about mathematics of learners or epistemological fraud. Based on these characteristics within the complexity cloud, the learner will reach some solution when solving problems, because of the nature of Mathematics and Science subjects. The solution can be either authentic or flawed, which feeds back into the schema, thereby building structure of the learner.

Papert (2005:354) introduced the term "Pop-Ed Culture", which points to the modern ideas about education and the mind as depicted in [Figure 2.21](#). Pop-Ed thinking considers:

- Blank-Mind theories, where a learner is conditioned to keep his/her mind blank in order to memorise something and wait for the solution to arrive instead of fetching the solution
- Getting-It theories, where a learner only recognises two states of his/her mind – either understanding or not understanding at all. The learner does not even try to take the necessary steps to overcome his/her lack of understanding if it cannot be accomplished in one 'bite'
- Faculty theories where a learner uses an extensive description or classification of the status of the mind or understanding of a concept, such as building a habit of

describing the learner as “he’s a brain” or “he’s a retard” or “I am not mathematical-minded”, instead of trying to analyse and diagnosing the shortcoming to overcome the problem or lack of understanding (Papert, 1980:355)

- This may give rise to a flawed solution (D’Amore, 2008; Jankvist & Niss, 2018) of the problem under investigation, as a learner is governed by theories that prevent him/her from developing a mathematical understanding of mathematical concepts
- Computational thinking, on the other hand, consists of the analysis, synthesis and concretisation of mathematical concepts. Computational thinking seems to be suppressed by the culture of Pop-Ed thinking (Papert, 2005) and epistemological fraud (D’Amore, 2008), as also found among learners in the sample group of this research. The probability of a correct result of the computational thinking component interaction is likely to be an authentic solution to the problem or striving towards an authentic solution. The solution, whether flawed or authentic, goes back to the learner, who uses the flawed or authentic solution in scaffolding new mathematical concepts building towards a schema. A typical example found among some of the learners who partook in this research, was that they simply did not know the sum of the angles of a triangle and guessed 60 degrees or 100 degrees. Such answers illustrate that learners have several answers for certain mathematical concepts that should have been understood as one correct answer only, based on the mathematical concept definition

Computational thinking consists of thought processes, abstraction and decomposition that are necessary to build and acquire the understanding of mathematical concept definitions (Selby & Woollard, 2014). From literature that was reviewed, the researcher found that a programming language may have the potential to raise the cognitive levels of formal operations of learners, thus paving the way for successful completion of tertiary courses involving Mathematics and Computer Science/Information Technology, instead of encouragement only. Through a programming language an embodied cognitive experience is evident, owing to more senses that are involved through practicing computational thinking. Even Tall (2003, 2004, 2008) describes the three worlds of mathematics as illustrated in [Figure 2.6](#). When arguing the importance of worlds 1 and 2, a programming language such as Greenfoot can strengthen the first two worlds without the learner having to become a specialist programmer. The third world of the “formal-axiomatic” (Tall, 2003:4), where logical deductions are made to prove theorems, may invade the formal world of a programmer where algorithms are figured out and debated, based on the pre-knowledge of the first two worlds in mathematics.

The Greenfoot programming language may provide a bridge to the specialised OO Java programming language, which benefits the private schools and not the state schools. The state schools abandoned Java as programming language driving IT as a subject in schools and replaced Java with a proprietary programming language. The reason for this is not clear to the researcher; clarification by the DBE is needed. However, research done by Goosen, Mentz and Nieuwoudt (2007) in South African schools contradicts this decision made by the DBE. The authors stated that affordability of the Java language was acceptable owing to a free ownership, as found by their significant sample used. The research was done when Java was still prescribed for both private and state schools before the decision of the DBE to phase out Java. OOP was regarded as a significant role player at the time, providing instructional value, critical thinking skills, analysis of programming problems and formulating solutions, and basic programming principles. Important programming language aspects such as data abstraction, Internet programming and design principles, emphasised in this research, were seemingly not important. Goosen, Mentz and Nieuwoudt (2007) used a sample consisting of educators as policy makers, trainers and teachers at tertiary institutions in SA to research Java usage and implementation among schools. The danger of using such research as a measure to judge Java as the best programming language in South African schools may be that the sample chosen voices a collective opinion from a subjective perspective of what is known to the sample only and that industry should play an important role in such a study. The future of a country does not depend on this country being the consumers of the developed countries only, as highlighted by Mpofu and Nicolaidis (2019), when considering the implications of 4IR. This research was done at a private school where Java was used for their IT subjects and where Greenfoot provided a natural language and a bridge towards Java. Java forms the foundation of Greenfoot. Portnoff (2018) states, that learning a programming language should be like language learning. Students will struggle when “English” is done in the first term and then “Russian” in the second term and so on. The focus should be on learner confidence to create a contingency of the programming language rather than jumping about teaching different programming languages. A specific programming language should be mastered properly by learners, instead of introducing different programming languages that may cause confusion among learners. These tendencies pointed out by Portnoff (2018) might also be the reason why the private school where this research was done, had their highest intake in IT learners from grades 10 to 12 following this research, which provided an introduction of the Java language to learners, prior to starting with pure Java in Grade 10.

Being providers of technology and not only consumers of technology may be achieved through constructionism, when the learner is an information constructor. Learners create their own subjective representation of objective reality. Learners construct their own

understanding when making connections between intrinsic knowledge and new external knowledge (McPhail, 2016). Sweller et al. (2019:264) state that intrinsic cognitive load is determined by the complexity of the information as well as the knowledge of the person processing the knowledge.

The word “communicate” emphasises the importance of natural languages, as found by Portnoff (2018:34), who conducted research on functional Magnetic Resonance Imaging (fMRI) in a 2014 study, that the comprehension of computer programs occurs in the same regions of the brain that process natural languages. The author also attaches language which becomes increasingly more sophisticated in describing our physical and mental perceptions, incorporating internal conceptions, which involves visio-spatial imagery. Programming requires a spoken language to describe and understand the problem, finding an appropriate solution or algorithm for the problem and a programming language to express the algorithm within such a computational notation.

This research developed research-based solutions for a “wicked” educational problem. This was achieved through analysis of the results from Educational Design Research, contributing to the body of scientific knowledge in line with the Plomp (2013) and the van Wyk and de Villiers’s (2018) model on EDR.

The way learners interact with mathematics is a concern as it does not deliver positive outcomes (Spaull, 2013). The outcomes of this literature study show that the mathematical approach in schools is governed by working through papers and examples, which may strengthen a concept-image approach outcome (Reddy et al., 2012; Spaull, 2013; CDE, 2014; Reddy et al., 2015). The approach may also re-iterate an incorrect understanding of mathematical concepts, because, according to Higgins and Wiest (2006), practice does not necessarily make perfect. Higgins and Wiest (2006) conclude that the learner may, by practicing a relative or warped solution, create a concept-image of a mathematical concept through a relative epistemological approach. The constructive approach may be more damaging than a guided approach, as the constructive approach generally allows the learner to explore the mathematical problem without guidance to discover a solution on his/her own. This often leads to a warped or relative idea of any mathematical concept, thereby affecting the epistemological views of mathematical education. It is argued that worked examples are important to minimise time spent on concepts. A worked example (Clark, Kirschner & Sweller, 2010, 2012; McPhail, 2016) usually originates from the professional who has the right answer; it is not based on a relative viewpoint of the truth from a learner’s perspective.

Mathematical solutions to a mathematical problem may seem correct until the answer proves differently and is thus relative when seen from the learner’s frame of mind. The student still

has to abide by the APOS framework, and solving a problem using a programming language such as Greenfoot offers a trusted solution immediately visible to the student, underpinned by the APOS theory. The APOS theory and a theoretical conceptual framework (Figure 2.21) were used to implement a practical approach as a validation study (Plomp, 2013), which points to Design-Based Research (DBR). DBR is used to validate the APOS theory against the conceptual theoretical framework.

2.2.4 The viewpoint of educators and professionals on programming

Programming literacy should be taught in schools (Papert, 2005; Wright, Rich & Leatham, 2012). Information transfer is not a viable format of education to ensure that learners are employed in the future (Gleason, 2018). The DBE as policymaker may provide more instances in interfacing programming within the curriculum. This can be achieved in focussing on what must be taught or achieved, instead of why a programming language should be taught. Maybe the support of the importance of programming literacy in the curriculum should be debated among a wider audience and not within the educational space only. A trigger for such a debate may be the 4IR facing education and industry in RSA. The problems that occur, such as low matriculation marks for mathematics (Spaull, 2013; Reddy, 2014; CDE, 2014; Reddy et al., 2015; Voogt et al., 2015) and science, enjoyed, at the time of this writing, widespread attention, which ties in with the 4IR. Papert (2005) sees the problem as one of assimilation and accommodation, referring to schema building.

The WCED introduced mathematics standard and higher grades at first. When learners could still not succeed in acquiring the specified mathematical skills, the concept of mathematics literacy was introduced. Papert (1980:38) describes this phenomenon as “Mathophobia” or the fear of learning. Education then confuses this “fear of learning” with poor aptitude (Papert, 1980:44). For this research, the researcher had to consider the positioning of mathematics literacy as an enabler of logical and abstract thinking, as in the case of mathematics. Bule and Seith (2012) opine that too much emphasis is placed on the availability of computers at schools, instead on teaching programming using a programming language. Bule and Seith (2012) further state that Livingstone, a computer games entrepreneur, refers to the “narrowness” of teachings about computers, creating digital illiterates in the UK, slowly killing important industries of the talent they need to maintain a competitive advantage. According to Bule and Seith (2012:1), Livingstone describes programming as the “lingua franca of competitive, innovative business”.

Papert (1980) argues that a language like LOGO blurs boundaries in such a way that no specific exercise is specifically allocated for learning mathematics or spelling proficiency, etc. (Papert, 1980). Bule and Seith (2012) further state that Scotland is delivering a higher

standard of computer programming at schools, involving lower levels of schooling, for PLs such as Scratch, Alice, and LOGO. The focus of Scotland is on solving the problem, instead of sorting the syntax. The authors further argue that with the advent of the Raspberry Pi, many instructors and stakeholders advocate a formal programming approach around teaching programming from as early as Grade 5, but there are many pedagogic arguments against the idea. Educationists advocate the pedagogic development of the child, which allows for the exposure of children to computer technology, but not to formal syntactical programming teachings (Bule & Seith, 2012). The researcher also supports this view, but as revealed by the literature review, dual-modality PLs such as Greenfoot may assist with the need. The aim of the BBC Microcomputer created by Prof Steve Furber of the University of Manchester underlines the teaching of PLs to create the opportunity to expose every child to programming (Bule & Seith, 2012).

According to Hartley and Treagust (2014), learners appreciate the use of computers in collaboration with their mathematics lessons. It was pointed out that these computers could only be used once or twice and that the computers are also shared, which may cause a breakdown in availability. The overall opinion is that if the situation was ideal, it would contribute to a better understanding, provided that the exercises do have a direct correlation with mathematics skills done in the classroom. It still remains the responsibility of both teacher and learner to ensure that the usage of computer programs to promote mathematics learning, correlates.

Papert studied under Piaget and much of this research is motivated by his research, which developed from a number of studies done by Dubinsky. According to Feurzeig and Papert (2011), the teaching of PLs should form part of the normal academic progress in order to effectively reduce these formal barriers. The authors used Logo programming at the time to introduce children to formal thinking processes in a playful manner. Feurzeig and Papert (2011) regard the constructivist vision in mathematics teachings as the active construction of knowledge. It is about creating an artefact based on a sound ontology (livari, 2007). This can be achieved by using a programming language and by reflecting, discussing even failed procedures to examine, analysing and repairing these failed procedures. The epistemology, according to (livari, 2007), for Design Research focuses on three types of knowledge, which are conceptual knowledge with no truth value, descriptive knowledge with a truth value, and prescriptive knowledge with no truth value.

Mathematics has many formal methods, which forces a learner to think about the problem in an uncontrolled manner. By using a programming language, many of these formal methods can be changed from abstract to concrete (Feurzeig & Papert, 2011). The thinking may become controlled thinking as opposed to uncontrolled thinking. Problem-solving and formal

concepts in algebra are major stumbling blocks for many learners and programming languages can be used in such a way that those hurdles become enjoyable challenges. Papert (2005) emphasises the processes of assimilation and accommodation at the core of Piaget's theory of development. According to the author, these two processes are seldom understood by educators, i.e., assimilation refers to when new ideas need to be reconstituted to fit the child's mental structures, using existing schemas. Accommodation happens when the interaction of many of these new ideas are assimilated into the child's mental structures causing change eventually (Papert, 2005).

Numerous studies abroad have been conducted on the concept of abstraction as a prerequisite for mathematics, but many of these studies were conducted at university level where students already acquired a level of abstraction within certain disciplines. In SA, Brijlall and Ndlovu (2013:14) also studied the "mental constructions during optimisation problems in Calculus" of high school learners based on the APOS theory, but it shows a state of being after instruction and at the final stages of the learner's schooling career. APOS theory focuses on a developmental perspective or is used as an analytical evaluative tool (Arnon et al., 2014), but the main focus is to understand how learners learn mathematical concepts. It is thus a constructivist approach. What this research focused on is the APOS concept before development or analysis can take place. The development of the individual or student at a cognitive level of formal operations is considered as an important stage of APOS when Piaget's stage of formal operations is in its initial phase, i.e. 11 year olds, to embrace abstraction as a necessary skill in mathematics and other disciplines that requires some form of abstraction. A much broader term, describing problem solving skills, encompassing abstraction and automation as its two main pillars, is computational thinking. Research has been done internationally on APOS analysis, but the research does not guide the teacher in understanding APOS as a way of thinking about the thought processes necessary for mathematics or computer programming (Dubinsky & Lewin, 1986; Dubinsky, 1991; Dubinsky & McDonald, 2001; Arnon et al., 2014; Brijlall & Ndlovu, 2013; Cetin & Dubinsky, 2017). The APOS theory is accepted as a theory already well-known to the reader, which in many cases is not the case. Bule and Seith (2012) argue that the over emphasis on ICT being the driving force behind computer education has led to "lessons on office products", which created a boredom, where learners are teaching themselves and not seeing a computing qualification in Information Technology as an option.

According to Bule and Seith (2012), Quinton Cutts (a senior lecturer at the University of Glasgow and an expert on Computer Science in education) wants Information Technology as a separate and independent discipline. Cutts also supports the notion that "learning to

program” teaches an individual a new way of thinking, which is invaluable to a “highly technologically-oriented world”.

The positioning of programming language education or education in information technology, evaluating the meaning of programming literacy, opens up research on computational thinking. Szlávi and Zsakó (2017) state that ICT competencies must be defined as part of a country’s national curriculum that encompasses many traditional IT and current ICT objectives, with algorithmic thinking as a central theme. The authors make mention of the South African National Curriculum Statement, which states that a learner must be able to design, implement, test and deliver efficient and effective solutions to problem situations. The statement is absolute, but the implementation of the statement may raise other questions. However, the debate on competency education is also an important facet raised by Denning (2017).

2.2.5 Target group

Grade 8 and 9 learners fall within Piaget’s cognitive level of formal operations (Piaget, 1964; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017). The majority of Grade 9 learners are faced with subject choices for Grade 10 that affect their future, as per DBE curriculum, and should make learners more responsible towards their studies. The complex multitude of factors such as language orientation, theoretical value belief, cognitive ability, and personality influenced by culture that may play a role, as pointed out by Cegielski and Hall (2006) (section 2.2.2.1(b)(ii), Figure 2.8), are partly ignored. Research in decolonising education in South Africa (Sayed, Motala & Hoffman, 2017) is a valid discussion, but may blur the focus of this research. This research focuses on natural and formal sciences that entail a programming language perspective with APOS theory as lens. Considering the complex multitude of other factors will direct the focus of this research away from the formal and natural sciences to social sciences. The researcher touch on beliefs about mathematics, but it becomes challenging and involved psychological research (Cegielski & Hall, 2006; Moscucci, 2007). Although Lee and Choi (2017) state that higher-order thinking is more directly affected by deep learning approaches than by epistemological beliefs or attitudes towards technology use, other researchers (Cegielski & Hall, 2006; Moscucci, 2007) state that epistemological beliefs affect academic performance and influence achievement motivation, which supports the outcomes of this research.

As an extra note, regarding research conducted by Vygotsky (1978), the researcher deals, among other things, with the fundamental role of social interactions in the development of children. His notion of the *zone of proximal development (ZPD)* describes the gap between what a child can achieve alone and what this child could potentially achieve with the help and

guidance from someone more knowledgeable, skilled or experienced. Greenfoot is also a gaming platform on which learners create a game intuitively. Collaboration happened naturally; learners started to help each other in getting their Actor objects functioning properly to be able to engage in the game. This kept the participants motivated to complete their programming.

2.3 Literature review summary

Chapter 2 provides clarity as to why computational thinking lacks among learners at a cognitive level of formal operations within the context of the South African educational system, and how computational thinking can be strengthened among learners. This was accomplished by investigating the role of computational thinking concepts at a cognitive level of formal operations, followed by the discovery of a programming language that may promote computational thinking skills among learners where APOS theory and the programming language share common ground. The reason for choosing a programming language was that programming in itself exists within computational thinking, but a link was needed between mathematical thinking and the chosen programming language to align the thinking processes of the learner with that of learners doing mathematics within the APOS framework.

The literature review supports the problem statement and research questions in the following manner:

- i) Piaget's cognitive level of formal operations prescribes development at the specific starting age of 11 years.
- ii) Embodiment plays a vital role in mathematics education, but also in any learning requiring computational thinking that forms part of cognitive development at a level of formal operations.
- iii) Constructivist and constructionist learning rely on "met-befores", which may be prone to a relative truth. The emphasis should be on constructing knowledge as opposed to constructivist approach.
- iv) Computational thinking does have its origin in Computer Sciences and is grounded in abstraction and automation.
- v) Abstraction is broken down in empirical, pseudo-empirical and reflective abstraction, where reflective abstraction forms the basis of mathematics learning and has a strong connection with computational thinking.
- vi) Abstraction is defined in different ways by different researchers, but APOS theory, which is also a framework and model, provides a clear path as to how the theory can be applied in mathematics learning.

- vii) Any learning framework proposed for mathematics must have the functionality of a model to be implemented as a methodology in order to guide learners with changing their line of thought and belief system.
- viii) Many PLs are available that provide different ways a learner can interact while still practicing computational thinking in general. Considering embodiment and the use of language as vehicles for mathematical learning, Greenfoot is an appropriate choice to house the features necessary to accomplish computational thinking as well as the visual component combined with a language of expression that satisfies APOS theory.
- ix) The success of a proposed solution for learners to understand concepts in creating or building schemata at a cognitive level of formal operations is governed by the process of Piagetian equilibration through organisation and adaptation.
- x) The term literacy was redefined using a programming language and computational thinking as an extension to the traditional definition of literacy.
- xi) Cognitive ability is not the only component of any individual's make-up, but cultural and social issues are also contributors to computational thinking.

The 11 points highlighted in the literature review hone in on how APOS theory may contribute to computational thinking within the Greenfoot programming language. From the literature review, it is unlikely that traditional mathematical learning does not satisfy the goals of the education departments in South Africa.

This triggers an investigation into how to strengthen computational thinking among learners and raises more “how” questions, as posed in RQ2. In section 2.2.2.1(a)(i), the cognitive theory levels are discussed. According to Piaget (1964), Young (2012), Cherry (2014), Ghazi et al. (2014), Barrouillet (2015), Bormanaki and Khoshhal (2017) the cognitive level of formal operations is the level where thought processes are prominent. Factors necessary to kick-start computational thinking at a cognitive level of formal operations are investigated through the SRQs. The study shows that mathematical problem solving and computational thinking are linked through the concept of reflective abstraction.

Reflective abstraction is used in the context of computational thinking (Cetin & Dubinsky, 2017). Denning (2017) describes Aho's (2012) definition of computational thinking as the thought processes necessary to formulate problems. Selby and Woollard (2014) also link thought processes to computational thinking. The thought processes should bring about solutions to problems. These solutions can be represented as computational steps and algorithms as depicted in Figure 2.10. Denning (2017) further argues that computation is a process consisting of a computational model together with computational thinking. An algorithm is a way to control any machine that uses the model. Aho (2012) states that

algorithms are implemented using a computational notation such as a programming language to create computational models. These computational models are abstractions at the core of computation and computational thinking. Selby and Woollard (2014) identified three aspects that are always found in the definition of computational thinking, namely, thought processes, the concept of abstraction, and the concept of decomposition. The authors further argue that the terms 'problem solving' and 'logical thinking' are too broad and focus more on skills development.

Abstraction and automation are the “mental and metal tools” of computational thinking (Wing, 2006, 2008:3718). Words and phrases such as “thinking at multiple levels of abstraction”, “decomposition”, “heuristic reasoning to discover a solution”, “prefetching and caching in anticipation of future use”, “recursive thinking”, and “algorithm and precondition” describe some of the skills needed to think like a computer scientist (Wing, 2006:33). Learners who master computational thinking are able to understand a relationship between subjects and activities within and outside of school (Philbin et al., 2013).

Denning (2017) posits that following any sequence of steps or algorithm does not necessarily make someone a computational thinker. Aho (2012) states that computational thinking is about finding appropriate models of computation to derive a solution for a formulated problem. Researchers such as Hayakawa (1949), Truran (1992), Wilensky (1991), Dubinsky (1991), Hazzan (1999, 2003), Devlin (2003), Kramer (2007), Perrenet (2010) and Meyer (2010) state subtle differences when arguing the concept of abstraction. Wilensky (1991:4) states “concreteness, then, is that property which measures the degree of our relatedness to the object, (the richness of our presentations, interactions, connections with the object), how close we are to it, if you will the quality of our relationship with the object”.

The APOS theory originated from Dubinsky's (1991) interpretation of Piaget's (1973) concept of reflective abstraction. Piaget sees the properties of objects not in the objects itself, but embedded in the actions taken by learners when they use these objects (Arnon et al., 2014). Each mental construction – **Action, Process, Object, Schema** (section 2.2.2.1(c)(vii), [Figure 2.18](#)) – or conception uses mental mechanisms (interiorisation, coordination, reversal, encapsulation and thematisation) to progress through the APOS (mental structures) cycle. Reflective abstraction is a description of what goes on in the minds of individuals when they are engaged in creating knowledge. It is hypothetical, as nobody can see what goes on inside another's mind (Dubinsky, 1991, 2000).

There are many possible ways to solve a mathematical problem, which may confuse the learners. This often leads to lower levels of abstraction that complicates their understanding (Hazzan, 1999; Kramer, 2007). Researchers describe this state a learner is in as a state of

“abstraction anxiety” (Sfard, 1991; Wilensky, 1991; Meyer, 2010), which forms an important component of mathematical anxiety. Papert (1980) uses the term “mathophobia” and Tall (2004) refers to this phenomenon as “dyscalculia”. According to Meyer (2010), educators should not only classify subjects as being abstract, but also deal with this anxiety associated with abstraction.

Many studies have been conducted on the concept of abstraction as prerequisite to subjects such as Mathematics and Programming (Wilensky, 1991; Dubinsky, 1991, 2000; Hazzan, 1999; Dubinsky & McDonald, 2001; Kramer, 2007; Perrenet, 2010; Meyer, 2010; Maharaj, 2013; Brijlall & Maharaj, 2014). However, many of these studies were conducted at university level where students already acquired the skill of abstraction in certain disciplines. Instruments for assessing certain characteristics associated with abstraction skills are then devised or used to measure abstraction (Hill et al., 2008; Perrenet, 2010). Unfortunately, only a few studies have been conducted to date at school level from grade 8 to grade 12.

2.3.1 Mathematics research

In researching journal articles from traditional English resources, researchers such as Bachelard (1938), Brousseau (1983), Piaget (1965), Duval (2006), Chevallard (2005, 2006), Op’t Eynde et al. (2002), D’Amore (2008), Vergnaud (1990, 2013) and Tramonti, Paneva-Marinova and Pavlov (2017) made contributions from a different perspective within the mathematics research discipline. The way learners approach mathematics vary in that some take a more difficult approach and others perform better, using the correct way, “... a different kind of mathematics that is often intolerably hard” (Gray & Tall, 1994:116). The US recognised the need through the National Council of Teachers of Mathematics (NCTM) Standards in 1989 to improve learners’ performance in mathematics. However, a well-informed group of stakeholders in education rejected the NCTM doctrine in search of “authentic reforms in mathematics education” (Budd et al., 2005:1).

Contrary to NCTM standards, the interest group against NCTM doctrine does not believe that:

- i) Learners must discover, but rather that discovery must be very selective in exceptional cases.
- ii) Learners must invent their own methods in performing basic operations, but should rather just study and practice the standard algorithms.
- iii) Learners must use a problem-solving approach and a drill-and-kill approach instead of practicing arithmetical operations. Learners will only remember what they practiced extensively.

- iv) Learners with learning abilities fit the curriculum better, but learners having learning disability perform better in a structured learning environment.
- v) Learners perform better using calculators that promote cognitive gains, but learners perform poor in calculus if they use a calculator in earlier grades.
- vi) Learners must be given work in context such as story problems to better understand concepts in context, but storyboard problems do not impact on understanding of future mathematics.

Considering [Figure 2.21](#), the theoretical conceptual framework (section [2.2.3](#)), is used as a starting point for this research based on the literature review. Within the concept of a flawed solution interpreted by the learner, McGowen and Tall (2010:170) advocate that it would not be “appropriate” to discuss the epistemological obstacles with a learner directly, but rather refer to the “met-befores” that led the learner to think in such a way. This is indicated in the interviews and helped with the learner’s reflection on his/her knowledge and understanding of mathematical concepts. This approach is also synonymous to the meta-belief system activity (MBSA) (Moscucci, 2007). The view of McGowen and Tall (2010) now shifts from an earlier “met-before” vision as advocated by the seminal author Tall (2003, 2004, 2008). It is necessary and important to understand a mathematical concept, to the idea that the “met-before” may also impede further learning. This finding contributed to the conjecture that the programming language provides a reliable “met-before”, because the outcome is tested with an immediate effect. Learners must therefore be taught in a way where concepts are questioned and problematic “met-befores” are minimised as learners deal with subjects like Mathematics and Science.

Other obstacles that face the learner are embedded in the usage of language and symbols, which play an important role in the understanding and execution of a solution to a problem. McGowen and Tall (2010) see this as ambiguity that may cause learners to interpret mathematical problems incorrectly. Within Greenfoot, deaf students may also engage in verifying the outcome of their coded algorithms as being truthful.

2.3.2 How does one involve the whole body?

The three mathematical worlds consist of the embodied, symbolic-proceptual, and formal-axiomatic worlds (Tall, 2004) ([Figure 2.5](#)). Our perceptions of the world consist of anyone’s thinking on how we perceive and sense things both physically and mentally. The author also attaches language which becomes increasingly more sophisticated in describing our physical and mental perceptions, incorporating internal conceptions, which involves visio-spatial imagery.

The other two worlds are augmenting the first world; hence, this research proposes the use of a visual programming language such as Greenfoot to allow a higher order of embodied cognition to take place. Greenfoot allows the learner to describe objects in an artificial world using actors to enact. These actors are physical, but the learner can strengthen their internal conceptions of visual-spatial imagery. Tall (2004:285) states that this will strengthen all geometries in “that can be conceptually embodied Euclidian and non-Euclidian geometries”. Within Greenfoot, the second and third worlds of symbols and properties can be addressed using process-object theories of Dubinsky (1991), with a controlled constructionist approach. Papavlasopoulou, Giannakos and Jaccheri (2019) describe these programming environments as child friendly. According to Portnoff (2018), languages such as Scratch and Alice may transfer to text-based languages.

2.3.3 Discovery learning creates challenges

Teaching new content and skills to novices, teachers should differentiate between learners. This calls for teaching and learning strategies for differentiation in a large class of learners, and it is challenging. Teachers need to walk learners through the procedure and concepts behind the procedure, at least in the initial phases of explaining how to solve a mathematical problem. Subsequent exercises may then have little or no procedural explanations. The authors see this partially guided approach as experiential learning and constructivist learning among other synonyms (Clark, Kirschner & Sweller, 2012).

Clark, Kirschner and Sweller (2012) state that many learners being taught through pure discovery methods, become lost and frustrated. According to the authors, research shows that those students mastering concepts using discovery learning inhibited no superior quality of learning. The authors further argue that only the brightest and well-prepared students succeed in making discoveries. Others do not participate or they simply mimic or duplicate the outcomes of the successful students. The worst case is when students discover a solution and it is a false interpretation or misinterpretation of the truth, also known as a relative epistemological outcome. This may influence further learning, especially in mathematics where certain concepts become actions and processes again to build new schemas. Of all the arguments put forward by the authors, the most important factor is the time it takes to teach a mathematical concept definition with constructivist approaches, which may increase by days instead of a normal 25-minute period.

The authors highlight an even more important concern in that learners will always choose the approach which has the least effect on their input to learn concepts. A less skilled learner will rather opt for a less-guided approach. According to Clark, Kirschner and Sweller (2012), a more guided approach requires of learners to provide a more attention-driven approach. On

the other hand, more skilled learners will opt for a guided approach, as it requires less attention and thinking. A constructivist approach is regarded as a means by which students should construct their own knowledge. Many educators propagate this as the discovery of knowledge in solving problems without explicit guidance. This is also known as a “constructivist teaching fallacy” (Clark, Kirschner & Sweller, 2012:8). The authors further argue that hiding or withholding information from learners cannot help with the construction of knowledge. There is a difference between “constructing knowledge” and a “constructivist approach”. The latter does not construct knowledge.

The brain learns through long term memory, short-term memory or working memory. Working memory can only hold information for a couple of seconds, unless the learner uses his/her long-term memory to fetch concepts previously learnt, such as a chess player who can scan multiple chess board moves to make an informed best choice regarding the move he/she needs to make. Long-term memory thus provides a holding area for a “worked-example” in mathematics. The learner can then, just like an expert, retrieve the worked-example from long term memory and successfully perform the procedure to solve a problem, based on that “worked-example”. Clark, Kirschner and Sweller (2012), also coined this as the “worked-example effect”. They see novices spending a considerable amount of time engaging in problem-solving activities and hardly learning anything during this engagement

This researcher investigated ways to improve mathematical skills by combining Dubinsky and Lewin (1986); Dubinsky (1991, 2000); Dubinsky and McDonald (2001); Tall (2008) research to apply programming language fluency.

2.3.4 Status quo of teaching and learning

Wilhelm (2008:45) points out that education in many instances entails low road transfer of learning, for real world problems are “ill-formed and there is always a remainder”. High road transfer of learning, which depends on “mindful abstraction”, is required in subjects that require computational thinking (Perkins & Salomon, 1992:8). The complexity of mathematics is also highlighted (Dienes, 1960; Bruner 1966; Biehler & Snowman, 1986; White, 2003; White & Sivitantes, 2002; Young, 2012; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017) as well as the theory behind the education in mathematics (Dienes, 1960; Piaget, 1965; Bruner, 1966; Dubinsky, 1991; Tall, 2004; 2008; Arnon et al., 2014). [Figure 2.21](#) shows the basic learner-subject interaction on computational thinking. In [Figure 2.21](#) the reasons for failures are depicted, such as the Pop-Ed thinking (Papert, 2005), which produce minimum authentic solutions compared to the majority of flawed solutions as highlighted in the Grade 12 (HSRC, 2014; DBE, 2015; Chirinda & Barmby,

2018) results every year. These flawed and authenticated solutions influence the Grade 12 results, which dictate career choices being made on performance.

However, the country needs certain career choices for its survival and economic growth, and performance lacks in those subjects that demand computational thinking. Learning now becomes a higher degree of embodied cognition. The next phase is the theory development phase, grounding the Educational Design Research (EDR) method.

CHAPTER 3: DESIGN RESEARCH

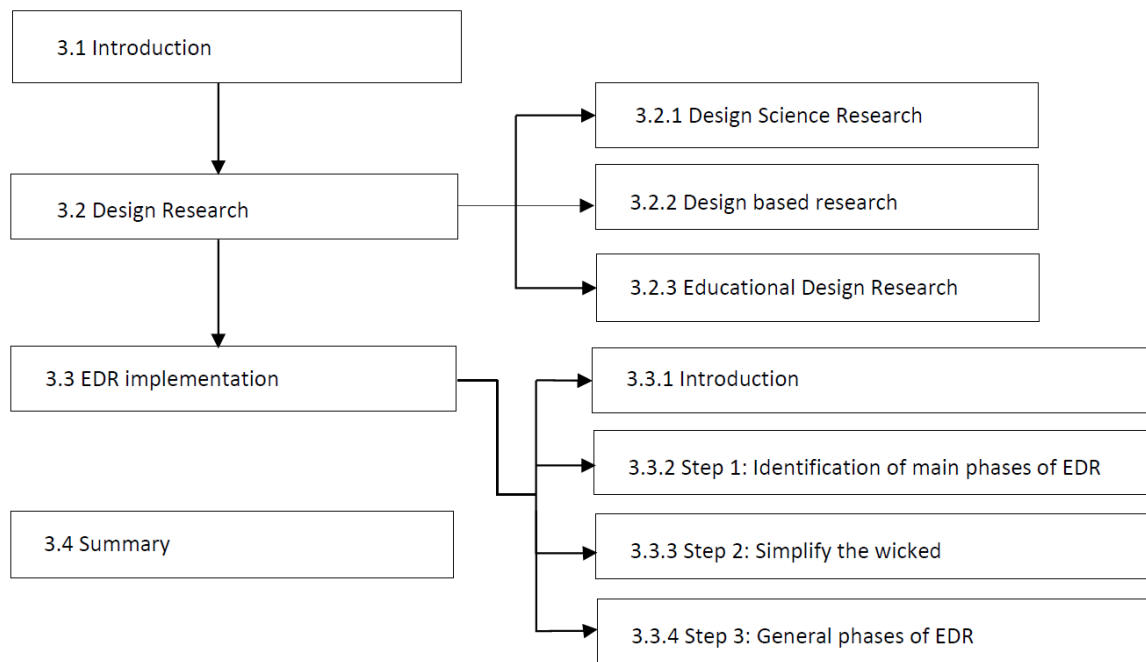


Figure 3.1: Layout of Chapter 3

3.1 Introduction

The Design Research (DR) was built on Design Science (DS) after discarding Action Research (AR) (section 4.5.2), and is discussed under headings covering design research (DR), Design Science Research (DSR) and Educational Design Research (EDR). Figure 3.1 represents the layout of Chapter 3.

3.2 Design Research (DR)

DR is applied DS, also known as DSR within the information systems discipline (Hevner et al., 2004). DR is described as research about design, and DSR is research using design as a research method (Vaishnavi, Kuechler & Petter, 2019). Good DR “must lead to shareable theories that help communicate relevant implications to practitioners and other educational designers” (Design-based Research Collective, 2003:5). Within education, DR is also known as Design Based Research (DBR), (section 4.5.2) or Educational Design Research (EDR) (section 5.3) (Van Wyk & De Villiers, 2018; Miah, Solomonides & Gammack, 2019).

A framework, as proposed by Venable (2006), on DR illustrated in Figure 3.2, suggests a solution as outcome. Venable (2006) consulted several contributors in DSR such as Nunamaker Jr., Chen and Purdin (1991), March and Smith (1995), Venable and Travis (1999) and Hevner et al. (2004), to create the framework. The solution points to any

workable technique in information systems, information technology, algorithms, and managerial practices, among others. It provides a detailed design, building and functional testing of a solution technology to provide or assist in the treatment or reduction of the “undesirable circumstances” of an identified problem (Venable, 2006:3).

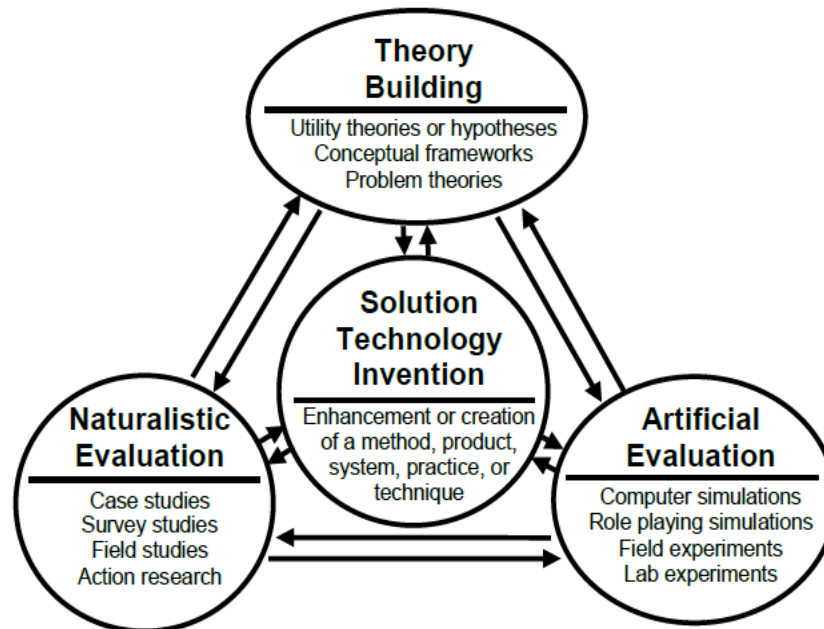


Figure 3.2: Framework and Context of DR (Adopted from Venable, 2006:3)

Venable (2006) further argues that the solution technology must be supported by theory building and tested through naturalistic or artificial evaluation. The naturalistic and artificial evaluation may borrow from each other. Artificial evaluation may include laboratory experiments, field experiments or simulations, whereas naturalistic evaluations include case or field studies, surveys, ethnography or action research. Naturalistic evaluation may be empirical, but people’s opinions or perceptions of whether the solution technology solved the problem, often carry more weight than an objectively verifiable phenomenon (Venable, 2006). In the next section DSR is discussed, followed by EDR.

3.2.1 Design Science Research (DSR)

Takeda et al. (1990) developed a design process model, which was adapted by Vaishnavi and Kuechler (2008) in terms of behavioural patterns for DSR, also known as “improvement research” (Järvinen, 2007:49; Vaishnavi, Kuechler & Petter, 2019:14). Vaishnavi, Kuechler and Petter (2019) adapted this model into a DSR process model based on the research done by Takeda et al. (1990), to describe cognition processes within the DSR process model (Figure 3.3).

According to Takeda et al. (1990), cognition is included in DSR through abduction, deduction and circumscription as cognitive processes. Venable (2006) and Pries-Heje, Baskerville and

Venable (2008) regard DSR as the evaluation of DS outputs, including theory and artefacts, where an artefact, according to March and Smith (1995:250), can be made up of “constructs, methods and instantiations”. According to Venable (2006:3), DR is more than DSR in that DR has theory building as “precursor and as a result”.

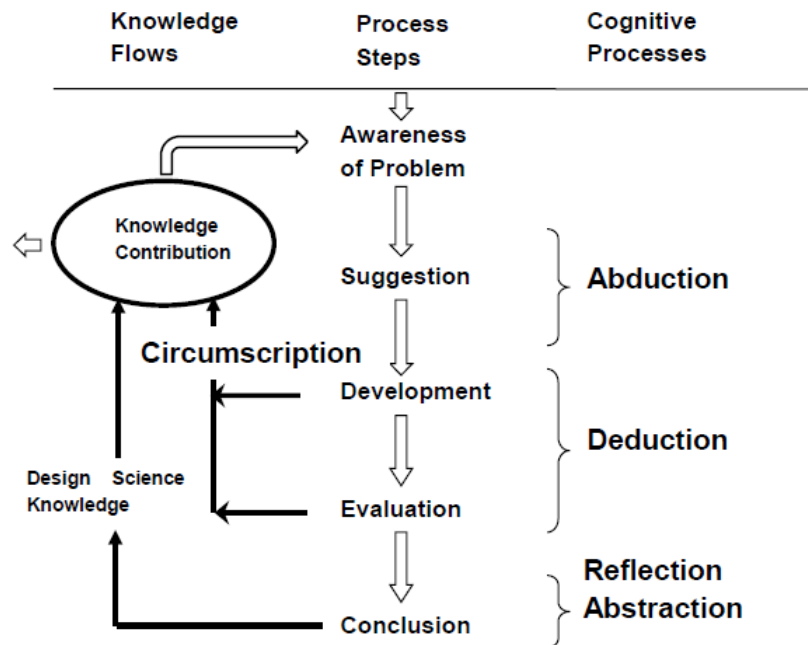


Figure 3.3: DSR process model (Adopted from Vaishnavi, Kuechler & Petter, 2019:14)

Winter (2008) defines DSR as reflection on, and guidance for the construction and evaluation process of an IT-artefact. Circumscription, as highlighted by Vaishnavi, Kuechler and Petter (2019) in Figure 3.3, forms part of the constraint knowledge, which is the process of maintaining knowledge about theories that did not work.

3.2.2 Design-Based Research (DBR)

According to Plomp (2013), DR in education can be conducted either as a development study or as a validation study in order to develop or validate theories. Plomp (2013) argues that Research-Based Design (RBD) points to development studies and Design-Based Research (DBR) points to validation studies where a theory is either developed or validated. Overall, whether developmental or validation studies, the purpose is to develop research-based solutions for ‘wicked’ educational problems through studying the educational interventions and so contributing to the scientific body of knowledge (Plomp, 2013). This is in congruence with the DR framework of Venable (2006) as depicted in Figure 3.2. This research focused on education and computer science (CS). EDR is discussed next as the DR/DBR option of choice for this research.

3.2.3 Educational Design Research (EDR)

3.2.3.1 Introduction

This research commenced with building the methodology around AR strategy (section 4.5.2), but the research strategy was changed to DR and, more specifically, to EDR. The EDR strategy suits the problem statement and research questions better. EDR is specifically used to solve a wicked problem in education and CS as discussed in section 3.3.3. The EDR strategy was followed, with the emphasis on exploring and explaining cognitive development through the teaching, learning and training of grade 8 and 9 learners (Bannan, 2013). According to van Wyk and de Villiers (2018), EDR has dual outcomes, namely, practical and theoretical contributions. This research produced an artefact of EDR as a real-world solution, and a proposed conceptual framework as the theoretical contribution in Figure 8.4.

3.2.3.2 Paradigms in EDR

The ontological stance of this research was subjective in the approach to explore and understand how the chosen programming language could be used to stimulate thought processes among learners in a similar manner to engaging in mathematical learning and computational thinking. In order to progress from the theoretical research to the evaluative and development research for this study, as illustrated in Figure 3.4, positivist and developmentalist paradigms had to be adopted, but this necessitated further research. The further research refers to allowing the adoption of this research based on scientific evidence, by a wider audience, such as the teachers of several school districts.

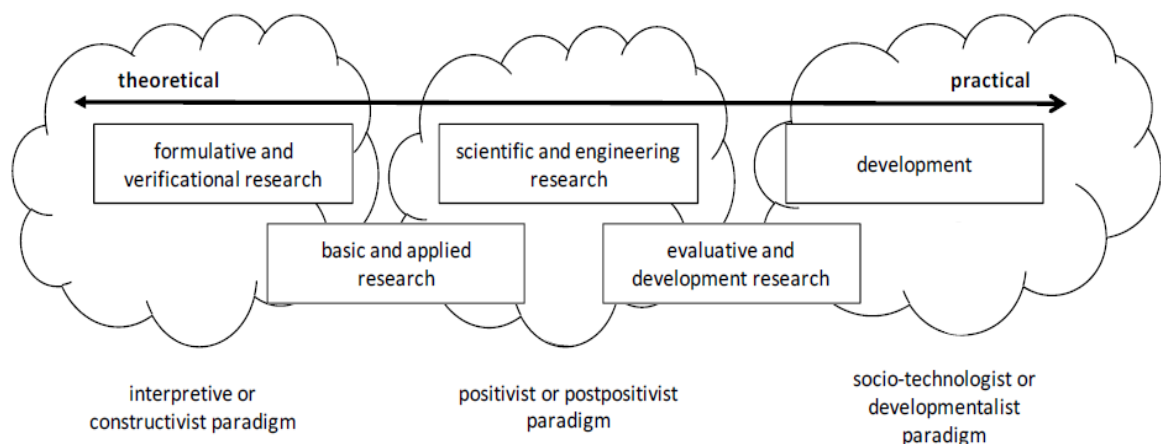


Figure 3.4: Perceptions of EDR objectives and methods (Adopted from Weber, 2010:4)

The actual completion of the full developmentalist process was not fully achieved in this research, which entails the evaluation of the IT artefact by the teacher community; it was only done up to the point of proposing or positioning an IT artefact, with the focus on theory-building goals within the interpretive paradigm (Weber, 2010). The EDR approach was

followed with the emphasis on exploring and understanding cognitive development through teaching, learning and training grade 8 and 9 learners (Bannan, 2013). The learners were from rich and poor communities, speaking different languages, and attending a private school.

The programming language Greenfoot (or computational notation) was used as an intervention tool. The study was conducted using APOS theory as lens. APOS is a proven model, theory and framework through which Mathematics, among other subjects, is mastered (Arnon et al., 2014).

3.2.3.3 EDR approaches

DR, and especially DBR, is used by an increasing number of researchers in artificial sciences. The EDR model of van Wyk and de Villiers (2018) was chosen for this research because it is an explicit model. Many researchers including Reeves (2006), Bannan (2013), Mckenney and Reeves (2012), Plomp (2013) and Miah, Solomonides and Gammack (2019) are improving and proposing models/frameworks to suit various DBR studies.

DBR encompasses different approaches that are determined by the focus areas of these studies. Applications of DBR/EDR studies are found in the research of Norwich and Ylonen (2015) that focused on learners with learning difficulties. The authors used a technique called “Lesson Study”, which they performed as action research to deliver an artefact. Where they have specific iterations to improve the area of focus and pedagogy, a DBR approach also allows the researcher to return to previous stages, which shows similarity in research techniques. Li and Chu (2018) used a DBR approach to investigate teaching and learning among Chinese learners. Wolcott et al. (2019) used DBR to introduce theories into pharmacy education in order to improve teaching and learning practices. Koivisto et al. (2018) use DBR to create models to educate simulation facilitators. Miah, Solomonides and Gammack (2019) proposed a general methodology for curriculum development, based on three phases, which are supported by the six components of Peffers et al. (2008). Van Wyk and de Villiers (2018) used EDR to develop, evaluate and improve two virtual reality safety training systems for the South African mining industry.

A brief overview of different EDR approaches is provided by Reeves (2006), Bannan (2013), Mckenney and Reeves (2012), Plomp (2013), van Wyk and de Villiers (2018) and Miah, Solomonides and Gammack (2019). Reeves (2006) illustrates the traditional DBR approach in [Figure 3.5](#) by emphasising a stronger connection between educational research and real-world problems in DBR. The diagrammatic illustration ([Figure 3.5](#)) reflects a generic illustration rather than a specific guideline.

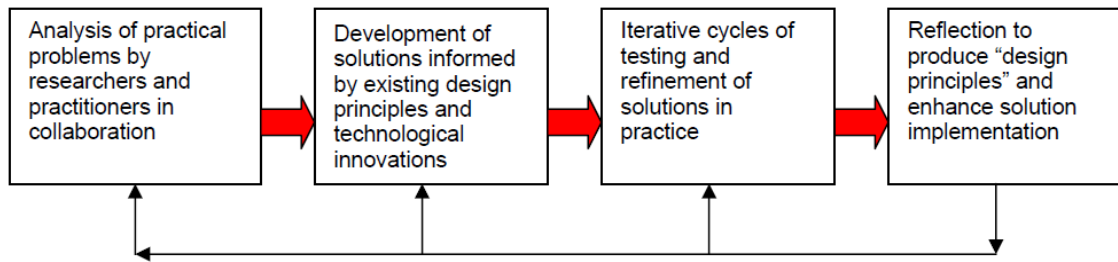


Figure 3.5: Refinement of problems, solutions, methods and design principles (Adopted from Reeves, 2006:14)

McKenney and Reeves (2012) proposed a generic model, depicted in Figure 3.5, with a generic approach. The model shows movement towards implied outcomes through reflection. The flow of activity may be in both directions, which is a more relaxed and open model compared to the model of Reeves (2006), as the analysis and design can be revisited.

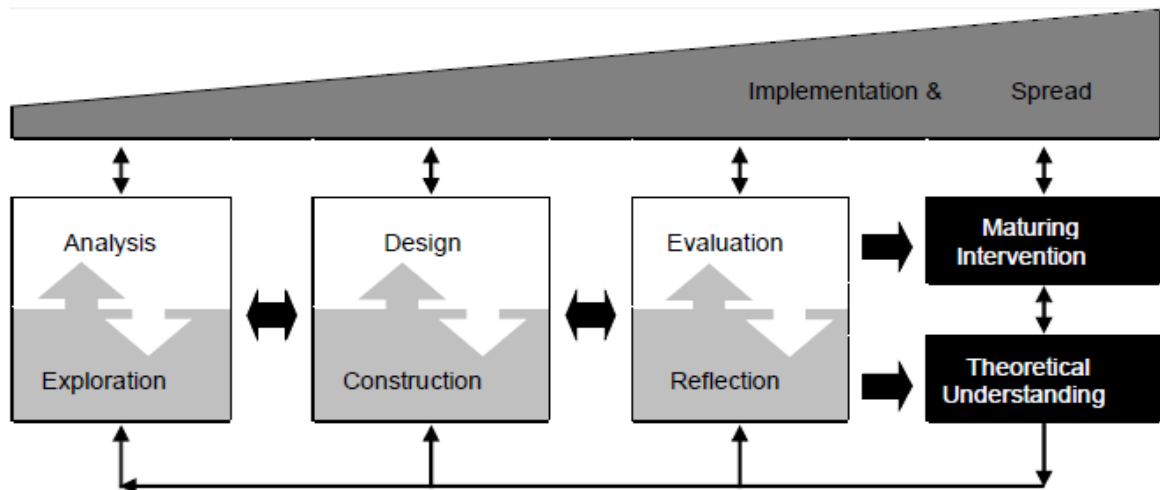


Figure 3.6: Generic model for EDR (Adopted from McKenney & Reeves, 2012:14)

Bannan (2013), on the other hand, uses the Integrated Learning Design Framework (ILDF) (Figure 3.7). The framework is quite extensive and may be overwhelming for researchers who are not familiar with EDR. However, the role of expert validation is included into this framework as part of the evaluation for general acceptance, complying with the developmentalist paradigm that necessitates a positivistic approach. Above all, the pragmatic demands of the learning environment are evaluated as part of the adoption of the artefact.

DBR is becoming a popular research approach for teaching and learning research used by researchers to design educational artefacts. Peffers et al. (2008) provide six defined activities when conducting design studies. The six activity steps are: (i) identify problem; (ii) define solution objectives; (iii) design and development; (iv) demonstration; (v) evaluation; and (vi) communication.

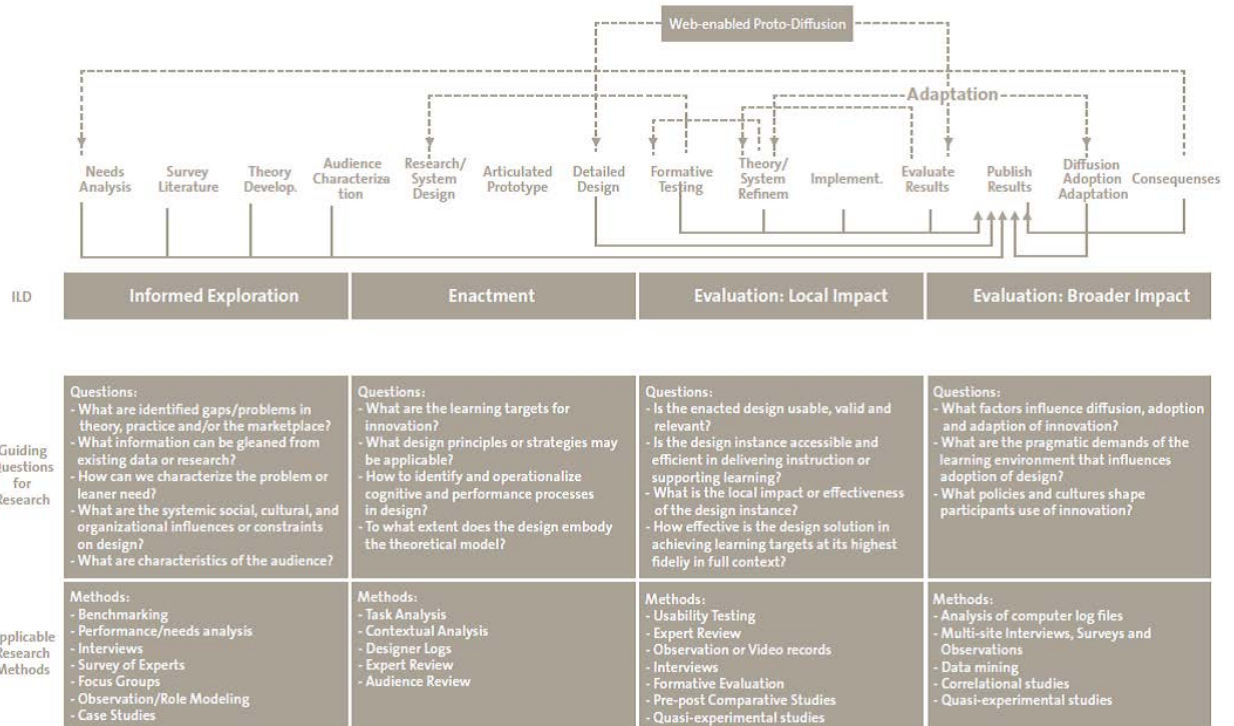


Figure 3.7: Questions and methods for DR using ILDF (Adopted from Bannan, 2013:55)

Miah, Solomonides and Gammack (2019) propose a three-phase DBR approach with iterations among these phases, built on Peffers et al.'s (2008) six activity steps. These iteration processes provide validity and relevance to the prototype or artefact. Although the authors emphasise communication to scholars, stakeholders and experts, this is not reflected in the methodology.

Lastly, van Wyk and de Villiers (2018) proposed a generic EDR model that integrates the phases and processes of the precedents (DR, DSR, DBR), as illustrated in Figure 3.8. All of the above approaches are based on iterative interventions within the proposed phases. The EDR model of van Wyk and de Villiers (2018) was chosen as the model for this research. This EDR model of choice is discussed in the next section.

3.2.3.4 EDR as model of choice

Having considered the different approaches, it has been concluded that they focus on three generic phases of a validation EDR study as proposed by Plomp (2013). These phases are: (i) preliminary research phase; (ii) prototyping phase; and (iii) assessment phase.

As discussed earlier, van Wyk and de Villiers (2018) proposed a generic model that integrates the phases and processes of the precedents (DR, DSR, DBR), as illustrated in Figure 4.8 (section 4.5.2), indicating the outcomes of each phase and not only in the last reflection stage as illustrated in models and frameworks of other researchers. The researcher

may revert back to any process if the intervention dictates such action in an iterative manner. Plomp's (2013) three-phase generic approach was used to subdivide the model of van Wyk and de Villiers (2018) in order to show pockets of development.

Vaishnavi, Kuechler and Petter (2019) emphasised circumscription and cognition. Van Wyk and de Villiers (2018) did not indicate these two concepts in their model. Circumscription and cognition should be added to the model to gain knowledge about interventions that were unsuccessful. The outcomes of the model as proposed by van Wyk and de Villiers (2018) were integrated under the three phases as described by Plomp (2013).

3.3 EDR implementation

3.3.1 Introduction

The EDR rollout was initially built on the Integrated Learning Design Framework (ILDF) as proposed by Bannan (2013), but the rollout was changed to the three generic phases of a validation EDR study as proposed by Plomp (2013) for simplicity. This research also demanded the focus to be primarily on: (i) coding using a programming language; (ii) LMS as educational approach for cognitive load theory; (iii) APOS theory being validated from a mathematics perspective; (iv) computational thinking, which encompasses portions of EDR and DSR to be included in this research. In terms of evaluation, the different EDR approaches are contained in Plomp's (2013) (Figure 3.9) phases, which are the (i) preliminary research phase, (ii) prototyping phase, and (iii) assessment phase, integrated into the model of van Wyk and de Villiers (2018). The authors largely focused on the outcomes of EDR, with outcomes for each process, which differs from other EDR models. The EDR approach adopted for this research can be classified as a validation study (Plomp, 2013), for the testing and application of the APOS theory as well as the design and evaluation of educational interventions. The van Wyk and de Villiers (2018) model is iterative and includes evaluation and reflection as integral stages in the model. The Framework for Evaluation in Design Science Research (FEDS) (Venable et al., 2016) was adopted for the evaluation phase. Although the steps followed in this research occurred before the release of Venable et al.'s (2016) paper, their paper gave structure to what was an intuitive approach during evaluation. The Gregor, Müller and Seidel (2013) framework was used for the reflection phase of this study. The reflection phase generated an EDR solution for the complex mathematical learning dilemma which points to a lack of computational thinking among high school learners in SA. To understand EDR as a process, it was subdivided into three steps, namely: (i) Identification of the main phases of EDR; (ii) simplifying the wicked problem to describe the artefact as outcome; and (iii) the implementation of an EDR approach.

3.3.2 Step 1: Identification of the main phases of EDR

This research used Plomp's (2013) generic phases of EDR as an abstracted overview of van Wyk and de Villiers's (2018) model as depicted in Figure 3.8. These phases are: (i) preliminary research phase; (ii) prototyping phase; and (iii) assessment phase. The outcomes of the synthesised model for EDR are manifested in the goals, initial design, artefact/prototype, research findings, and in the practical solution as well as the theoretical contribution, through reflection (Van Wyk & De Villiers, 2018).

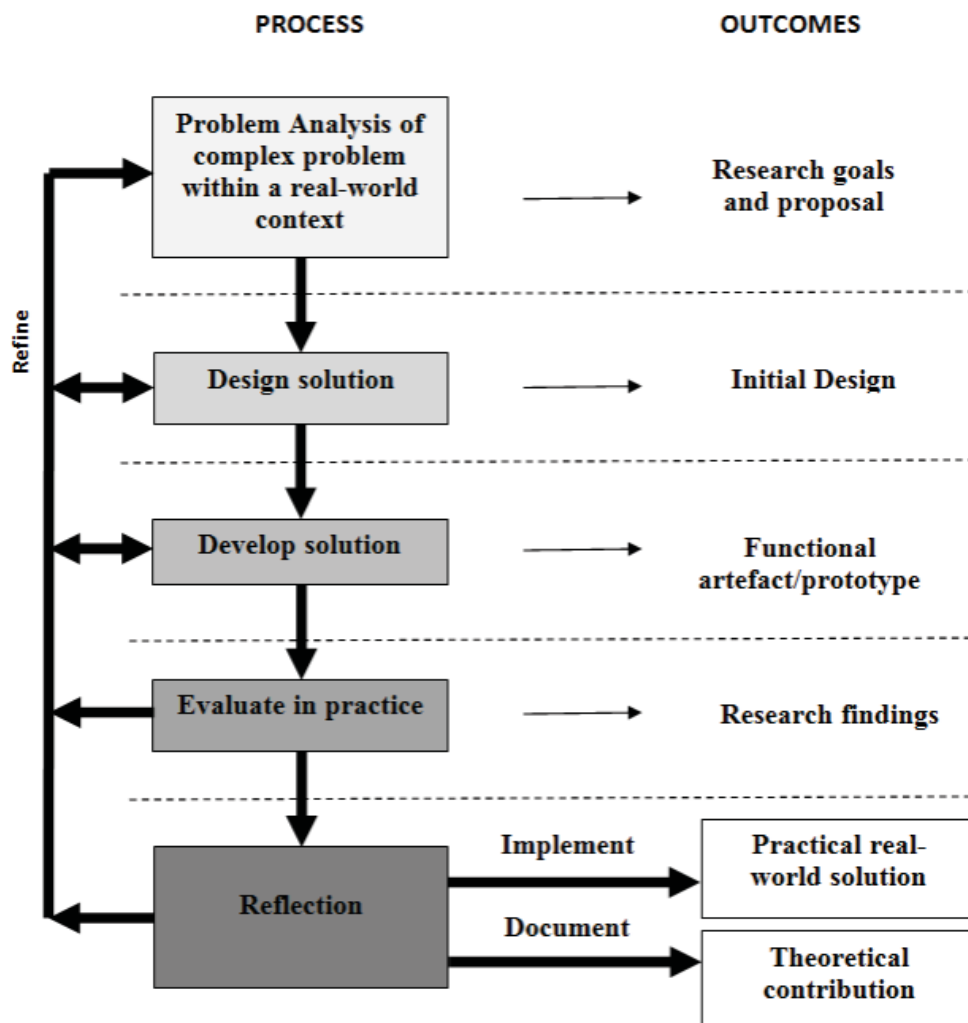


Figure 3.8: Synthesised generic model for educational DR (Adopted from Van Wyk & De Villiers, 2018:305)

3.3.3 Step 2: Simplify the wicked problem

The wicked problem in education must be investigated to make it less wicked. All the properties of a wicked problem are embedded in this research (Rittel & Webber, 1973; Camillus, 2008; Peters, 2017) namely:

- i) The content knowledge is fragmented into mathematics, computational thinking, PLs, learning management systems and pedagogic principles.
- ii) The teaching of the content is not transparent to learners and teachers although instructional materials are available, but disconnected.
- iii) The teachers' knowledge and skills are substandard in many cases.
- iv) Ubiquitous nature of learning is an important add-on for the research.

3.3.3.1 EDR research question

For this research, the van den Akker (1999:9) design principles dictated the construction of the EDR research question. Furthermore, the EDR research question was structured to address the intervention for the APOS theory validation. EDR is specifically designed to solve wicked problems in education and produce the necessary outcomes through state-of-the-art knowledge. Every component that plays a role must be interconnected to satisfy consistency, and above all, the solution must be sustainable for the specific education community (Plomp, 2013).

Consideration was given to the current two research questions to create one EDR research question according to the Van den Akker (1999) EDR questioning framework to address interventions, namely:

“Design for Intervention X for the purpose/function Y in context Z, then you are best advised to give that intervention the characteristics A, B, and C [substantive emphasis], and to do that via procedures K, L and M [procedural emphasis], because of arguments P, Q and R” (Plomp, 2013:15). This design principle is also congruent with the didactic contract (Brousseau, Sarrazy & Novotná, 2014) and the didactic situation (Brousseau, 2010). The didactic situation is seen “where an agent, for example the teacher, organises an intervention that manifests its intention to modify the knowledge of another agent, or causes it to develop” (Pepin, 2014).

The second agent, for example, is the learner who is allowed to express him/herself through actions. Pepin (2014) further states that the didactic contract is what the learner expects from the teacher and what the teacher expects from the learner. The didactic contract refers to a system of rules that applies to learner and teacher. Substantive knowledge refers to the necessary characteristics of an intervention and procedural knowledge refers to a set of design activities to produce a workable prototype.

This research supported the didactic situation of Brousseau (2010), embedded in Van den Akker's (1999) design principles, as indicted in [Table 3.1](#).

Table 3.1: The structure of the EDR question

EDR questioning framework parameters	EDR question values
Design for Intervention X	The design for interventions within the research project to validate
For the purpose/function Y	The APOS theory
In context Z, to give that intervention(s)	In context of computational thinking, employing
The characteristics A, B, and C [substantive emphasis]	The characteristics of the programming language's scenarios, classes, methods and objects at a cognitive level of formal operations, accomplished
To do that (accomplish) via procedures K, L and M [procedural emphasis]	Through an LMS to uphold the cognitive load theory (CLT) of learners and programming using a visual interface, frames-based editor, progressing to syntax-coding and debugging
Because of the accomplishment of arguments P, Q and R	Owing to performing specific activities by design, such as: lectures, flipped classroom techniques, exercises (ACE), algorithmic design of problems, abstraction and online tests

The EDR question fulfils the didactical contract and focuses on promoting computational thinking through coding in a programming language using APOS theory as lens. Furthermore, a didactic transposition (Chevallard, 1989) points to the programming language as tool to be put to knowledge as an entity that is instructed to, and studied by learners. Chevallard (2006) describes praxeology as the way in which society creates knowledge in an organised way. Praxeology consists of praxis and logos (Bosch, Gascón & Trigueros, 2017). In context of this research, praxis points to tasks or problems that need an algorithm, using a technique through APOS theory, to promote computational thinking in order to create the algorithm mapped into Greenfoot. The logos part points to procedures such as lectures, algorithmic design of problems, abstraction, programming, exercises, online tests and flipped classroom techniques that are embedded within theory and technology respectively (Postelnicu, 2017).

The Van den Akker (2003) design principles are thus aligned with the research of Brousseau (2002) and Chevallard (2006). This research was rolled out based on a combination of EDR, programming language, Moodle and APOS. What started in EDR, affected actions in the programming language, Moodle and APOS. After combining the two research questions, the technological educational intervention was brought about by iterative formative evaluations (Van den Akker, 1999). The research question according to Van den Akker's (1999) design principle (Table 3.1) for EDR reads as follows: "What teaching and learning strategies can empower learners to mastering computational thinking skills, through APOS theory, which is expected to function at Piaget's cognitive level of formal operations, infused by concepts and

characteristics of a programming language at high schools, in order to cope with the challenges in subjects such as Mathematics and Science?”

3.3.4 Step 3: The general phases of EDR

The EDR phases (Plomp, 2013) are summarised in [Figure 3.9](#).

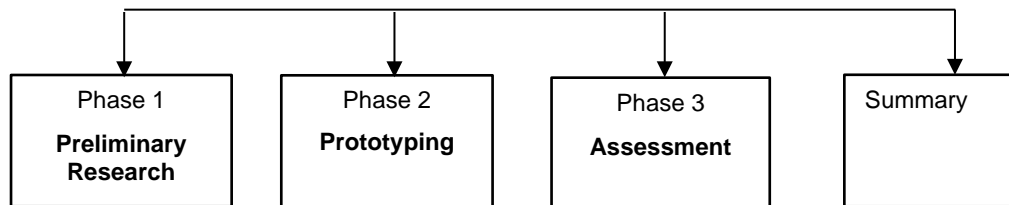


Figure 3.9: Overview of phases (Adopted from Plomp, 2013:19)

The preliminary research phase (Phase 1) required (i) a needs and context analysis, (ii) a literature review, (iii) a conceptual theoretical framework, (iv) and the identification of a target group ([Figure 3.10](#)).

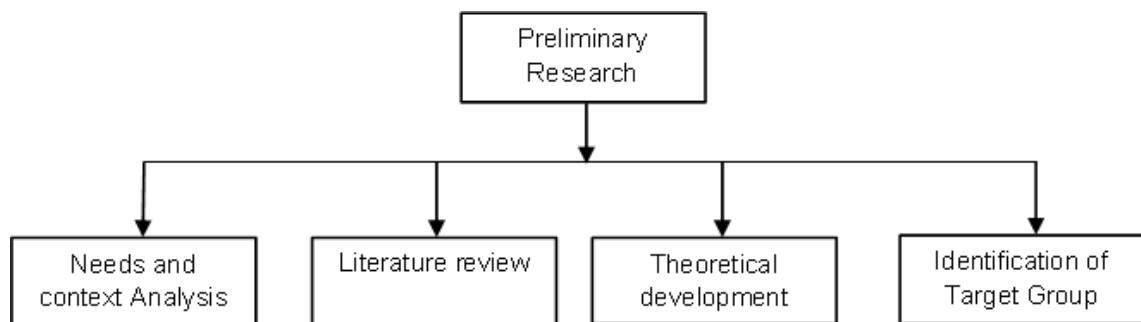


Figure 3.10: Diagram showing the problem identification phase (Adopted from Plomp, 2013:19)

Having the artefact specified within the EDR question, namely “the teaching and learning strategy to empower learners to master computational thinking”, the EDR phases were implemented. The outcomes of the van Wyk and de Villiers (2018:305) model were classified and integrated according to the proposed phases of Plomp (2013), namely:

3.3.4.1 Phase 1: Preliminary research

This phase consisted of an exploration of the problem through a needs and context analysis, a literature review, theory development, and target group identification that required preliminary research (section [5.4.2](#)). Within the model of van Wyk and de Villiers (2018), this phase points to the problem analysis within a real-world context. The problem is wicked, authentic and practical. Literature was reviewed on the wicked problem to generate relevant theory. The researcher collaborated with practitioners and set research goals, which generated a research proposal based on these goals. The problem analysis and

identification focused on making the wicked problem less wicked and understanding the problem logically. This is discussed in detail in Chapter 5 (section 5.4.2.1) under (i) needs and context analysis, (ii) literature review (section 5.4.2.2), (iii) theory development (section 5.4.2.3), and (iv) identification of the target group (section 5.4.2.4). This design was influenced by contextual limitations and the complexity of the interactions in real-world settings. Finally, a solution was designed that functioned as initial design, iteratively feeding back into the research goals and proposal (Van Wyk & De Villiers, 2018).

3.3.4.2 Phase 2: Prototyping

The approach of van Wyk and de Villiers (2018) was followed in developing a solution in the form of a prototype or artefact that fulfilled the research purpose. The design principles and technological innovations determined the process of development, which led to an innovative and functional artefact. It assumed the form of a construct, model, method or instantiation (March & Smith, 1995). During this phase, interventions were identified and supported (section 4.5.4) by adding tentative products and design principles (Wademan, 2005) to the intervention through formative evaluation in order to address this complex educational problem. Interventions are built on the theoretical conceptual framework and presentation mode of an intervention (Nieveen, 2013).

3.3.4.3 Phase 3: Assessment

Data were collected from participants and the findings were mapped against the research questions (section 5.4.5; Table 5.14). A theory was constructed according to the van Wyk and de Villiers (2018) model. The practicality of the theory was evaluated. Another essential component of the assessment was reflection. The researcher reflected on findings and analysed the findings to create categories and subsequent themes. Through reflection, the dual outcomes were presented as a practical real-world solution and a theoretical contribution in the form of a theory. The reflection on findings occurred through multiple EDR cycles until the process was exhausted.

Executing a research process that accommodated computational thinking within an existing curriculum was complex. The term 'technology-enhanced learning environment' (TELE) refers to the integration of technology into teaching and learning processes and may promote self-regulated learning (SRL) (Andrade & Bunker, 2010). In this research, the educational practices needed educational interventions that accommodated TELEs to promote higher-order thinking (Lee & Choi, 2017). Most interventions called for TELEs, which were either illustrated in the Moodle LMS integration to comply with cognitive load theory and a computational notation called Greenfoot as the programming language of choice, supported by the Moodle LMS. The educational practices supported by the educational interventions

improve the quality of learning, also known as TELE&T (Lee & Choi, 2017:144), where the “T”-suffix refers to teaching. The next section summarises Chapter 3.

3.4 Summary

Chapter 3 highlighted the foundation of DR. DR is viewed as DSR, DBR and EDR. The focus is on DR in education and the term DBR is used. Takeda et al. (1990) added cognition to DSR through abduction, deduction and circumscription, which also applies to DBR. DR in education can be conducted as RBD or DBR. DBR was the focus of this research to validate the APOS theory. EDR is more specific with describing DR in education that has dual outcomes, namely, as practical contribution, producing an artefact as a real-world solution, and as theoretical contribution, the proposed conceptual framework.

Having decided on EDR as the DR of choice, many approaches were considered. Most of the approaches were generic proposed models, which lacked specific outcomes. The EDR of choice was the van Wyk and de Villiers (2018) model. This model is descriptive and the three phases of Plomp (2013) are clearly visible within this model. However, Vaishnavi, Kuechler and Petter (2019) introduced circumscription and cognition to note the interventions that were unsuccessful. “Circumscription is a rule of conjecture” found in constraint knowledge about theories that are gathered through detection and analysis of contradictions when the theory does not apply (McCarthy, 1980:27; Kuechler & Vaishnavi, 2011; Vaishnavi, Kuechler & Petter, 2019:15).

The research design is discussed in Chapter 4 to elaborate on the context in which the research was conducted.

CHAPTER 4: RESEARCH DESIGN

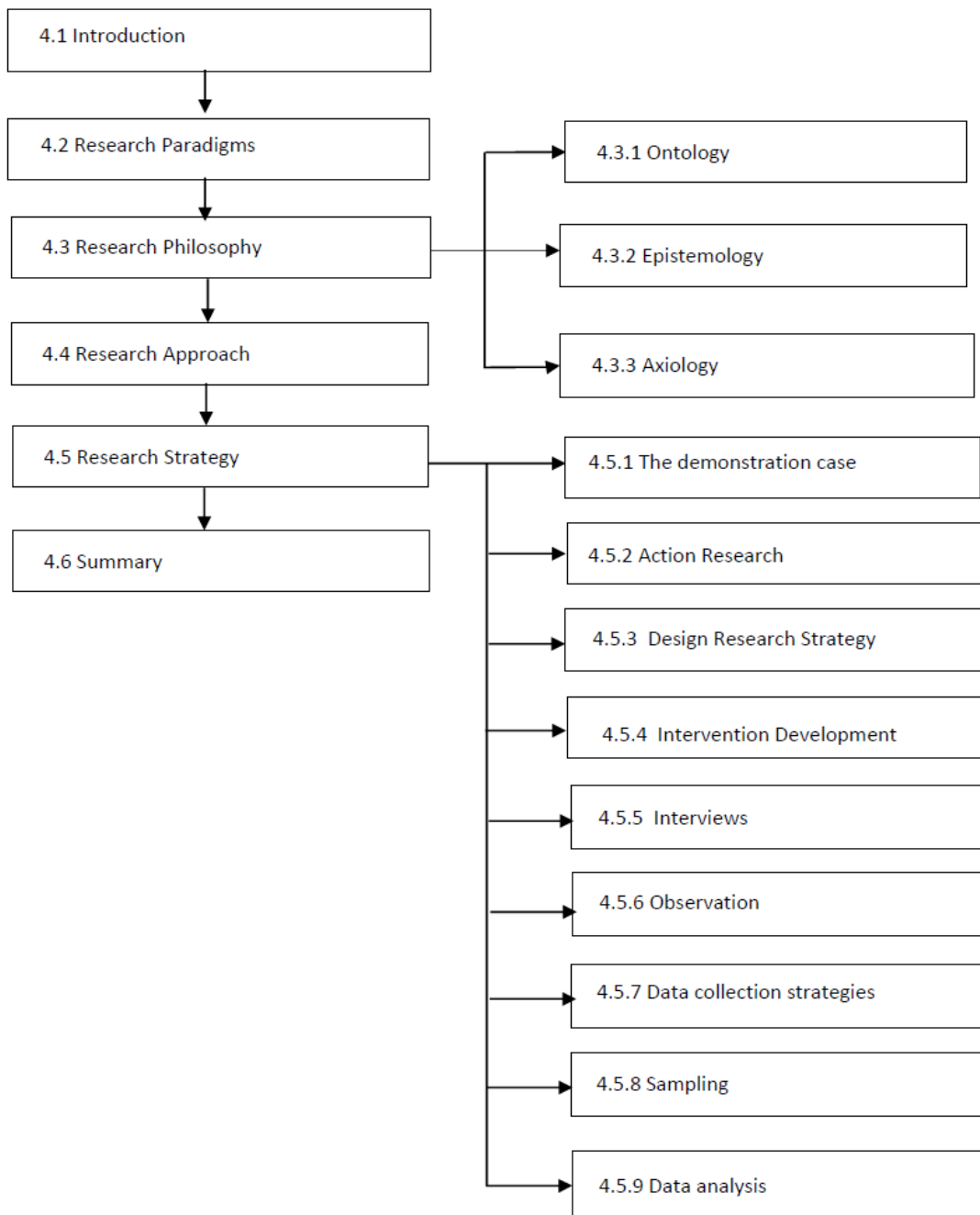


Figure 4.1: Layout of Chapter 4

4.1 Introduction

Research design forms the intersection of the research theory, research methodology and the context in which research is conducted (Jonker & Pennink, 2010). This chapter summarises the (i) research problem, (ii) research questions to address, (iii) theoretical conceptual framework, (iv) research methodology, (v) strategy used, (vi) data collection, and

(vii) analysis techniques and procedures. The terminologies are sourced from Saunders, Lewis and Thornhill (2019:128-170).

A sample of grade 8 and 9 learners was purposively drawn from a private school. Thirty-eight (38) learners formed two groups (15 and 13 learners respectively) that participated in the study. The data analysis was done sequentially. All the findings were summarised, categories were then grouped into themes. The findings were gathered during interventions (section 4.5.4) introduced to learners, and through assessments, tasks, analysing transcriptions of interviews and notes from observations (Appendix A-Z). Because of the nature of the wicked problem, a DR approach was adopted and an interpretivist-pragmatic approach followed. In practice, this meant that depending on the situation, the research fluctuated between interpretivist and pragmatic research.

The chapter is divided into the following sections, as indicated in [Figure 4.1](#): the research paradigms, philosophy, approach, strategy, and a summary of Chapter 4.

4.2 Research paradigms

Guba and Lincoln (1994:107) view a paradigm as “a set of basic beliefs”. It represents a worldview that defines the nature of the world as lived by the researcher and the relationships between the researcher and the world. When criticising the term “belief”, D’Amore’s (2008:3) definition states that a belief is “an opinion, set of judgements and of expectations, that which one thinks with regards to something”. The term “paradigm”, according to Lauffer (2011:49), is also used to describe a model or a conceptual framework, where a model is a representation of reality. A paradigm is a set of fundamental assumptions and beliefs in how the researcher perceives the world (Jonker & Pennink, 2010; Wahyuni, 2012). Wahyuni (2012) sees ontology and epistemology as philosophical dimensions to identify the research paradigms. Wahyuni (2012:69) argues that, “axiology and methodology are two beliefs which impact the epistemology paradigm pointing to either interpretivist or pragmatist research, where the epistemology is subjective and on social phenomena”.

According to Burrell and Morgan (1979) and Saunders, Lewis and Thornhill (2019), certain assumptions are made by the researcher which revolve around human knowledge (epistemology), the realities within the research (ontology), and how the researcher’s own values and beliefs influence this research process (axiology). These assumptions and basic beliefs define inquiry paradigms and are summarised by four questions namely:

- (i) Ontology: What is the form and nature of reality? (See section 4.3.1).
- (ii) Epistemology: What is the relationship between the knower or the candidate-knower and what can be known? (See section 4.3.2).

- (iii) Axiology: How will the research process be governed by the researcher’s values and beliefs? (See section 4.3.3).
- (iv) Methodology: How can the knower or candidate-knower extract findings that he or she believes can be known? The methodology of EDR is discussed in section 4.3.3.

According to Freshwater and Cahill (2012), all research methods fall within a paradigm resting on ontological, epistemological, axiological and philosophical groundings. Saunders, Lewis and Thornhill (2009) see a paradigm (functionalist, interpretive, radical structuralist and radical humanist) as a way to study social phenomena and through these studies, the researcher may understand and explain them better in stating his/her research philosophical position as positivism, critical realism, interpretivism, postmodernism or pragmatism.

This research starts in the radical humanist quadrant of the Burrell and Morgan (1979:22) model, based on “what” questions, conducted as an abstract subjective exploration of computational thinking (Cronje, 2016). The research then moves into the interpretivist and then functionalist quadrants sequentially. Burrell and Morgan (1979) state their proposed paradigms in Figure 4.2 for analysis of social theory as mutually exclusive. The authors further argue that it is possible to operate sequentially in different paradigms over time.

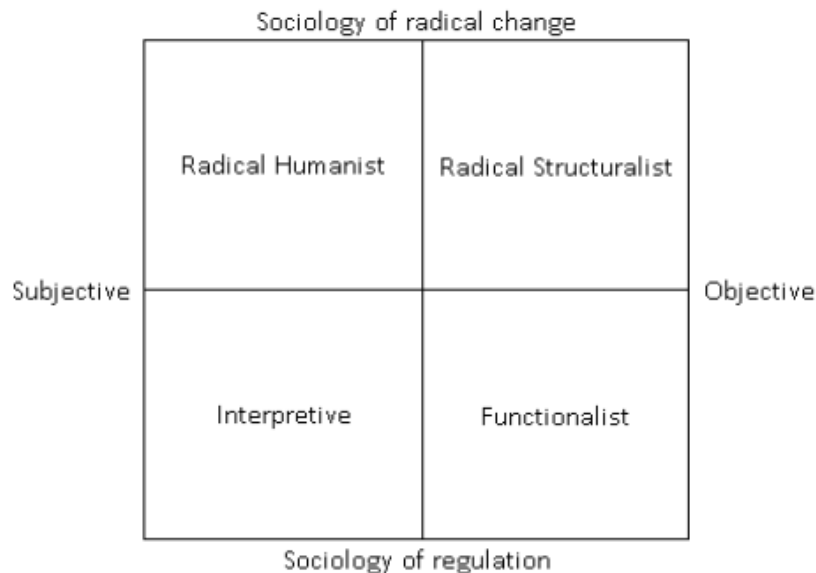


Figure 4.2: Four quadrants of sociological and organisational research (Adopted from Burrell & Morgan, 1979:22)

The purpose as set out in the aims of this research calls for radical change in such a manner that computational thinking skills may be made part of a learner’s schemata to be used in subjects such as Mathematics and Science. Computational thinking is explored by “what” questions (section 1.7; section 2.2.2.1), which produced a theoretical conceptual framework as outcome (section 2.3.2).

4.3 Research philosophy

The way researchers view the world determines “how” a researcher develops knowledge. The “how” is vested in the beliefs and assumptions of the researcher (Freshwater & Cahill, 2012; Saunders, Lewis & Thornhill, 2019). Saunders, Lewis and Thornhill (2019) further argue that research philosophies are investigated through research paradigms that points to the ideological orientation of the researcher, as highlighted by Jonker and Pennink (2010); Wahyuni (2012) and Burrell and Morgan (2016). For this research, the researcher investigated social phenomena and gained specific understanding of these phenomena to explain the findings. Saunders, Lewis and Thornhill (2019:130) view research philosophy as “a system of beliefs and assumptions about the development of knowledge”, as depicted in Figure 4.3. The authors further argue that research philosophy is a reflexive process because of too many variables and the assumptions of the researcher, all of which influence the research philosophy. When criticising the research onion relative to this research, the researcher adopted an interpretivist-pragmatist research philosophy by adding to the theory an abductive approach using a research strategy, i.e., EDR. Saunders, Lewis and Thornhill (2019) do not include DSR and EDR as a research strategy, as their focus is on business and management.

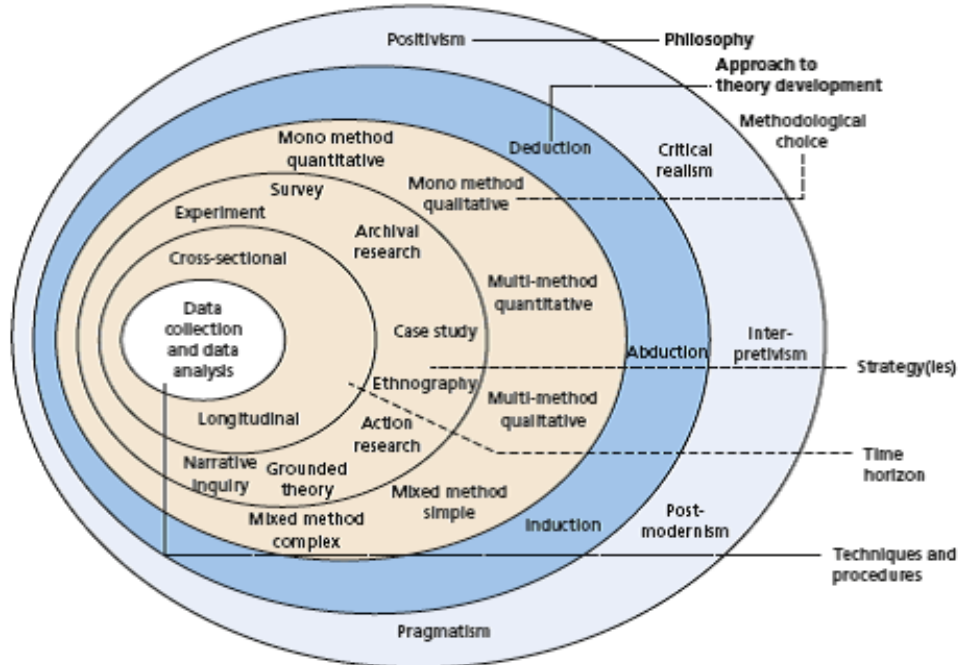


Figure 4.3: Research Onion (Adopted from Saunders, Lewis & Thornhill, 2019:130)

Answers to the research questions stated in Chapter 1 provide a deeper understanding of how programming language concepts may assist learners to achieve the cognitive level of formal operations, thereby sustaining computational thinking. This research aimed to explore

and understand how a programming language, using Action Process Object Schema (APOS) theory as lens, could promote computational thinking skills at a cognitive level of formal operations among high school learners (section 2.2.2). APOS is a proven model, theory and framework (Arnon et al., 2014) through which mathematics, among other subjects, is mastered. The usage of a programming language, namely Greenfoot, is highlighted and motivated in the literature review (section 2.2.2.1(b)) and tools within Greenfoot augment the APOS strategy. Research on the possible improvement of marks in mathematics by coding in a programming language is a topic for further research and not a direct outcome of this research. The focus of this research was not on mathematics *per se*, but on computational thinking through APOS within the Greenfoot programming language as computational notation.

4.3.1 Ontology – The nature of reality

According to Allison and Pomeroy (2000), ontology and epistemology are linked to values, hence the important debate around discovering the truth based on either objectivism or subjectivism. The social actors in this study were the grade 8 and 9 learners and the researcher/ teacher/ technologist. Multiple realities exist for each actor within social constructionism when reality is constructed “intersubjectively”. Several perceptions about logical thinking and cognitive development played an integral part of the social reality of education in this study. Objectivism, on the other hand, is a social reality external to, and independent of the social actors involved in the development of cognitive thinking as an example (Saunders, Lewis & Thornhill, 2019). For this study, such an approach seems less desirable, as the learner and researcher interacted and intervened with the specific interventions applied. The ontological stance is thus a subjective one. The researcher adopted an ontological stance, as the grade 8 and 9 learners contributed to the social phenomena.

4.3.2 Epistemology – The nature of knowledge

The nature of knowledge is also referred to as epistemology. Papert (1980) states that the nature of knowledge (epistemology) is not to describe the study of the conditions of validity of knowledge as is done by positivists, but rather to question the sources and growth of knowledge. Epistemology is the relationship between the knower and the known while one gets to know reality and discover truth in a subjective or objective manner (Kim & Donaldson, 2018). Allison and Pomeroy (2000:2) best explain epistemology as “what we do know and can know”. Muis (2004:317) regards epistemology as that branch of psychology involved in “the nature of knowledge and the justification of belief”. Epistemology is thus the relationship between the knower and the known while one gets to know reality and discover truth in a subjective or objective manner. Conducting research using DR, the epistemological

stance was adopted through interpretivism and pragmatism by taking action through cyclical interventions. The research moved between interpretivism and pragmatism, especially when the EDR methodology was applied. The researcher interpreted data pragmatically to develop new interventions as the research progressed. The focus of the research was on problems, practices and relevance during the application of APOS theory in a programming language to inform future practices such as computational thinking among high school learners.

Because the ontological underpinning of the study is subjective, the epistemological stance focused on interpretivism of the written, spoken and visual attributed meanings put forward by the opinions of learners, subjects or actors. The research was, above all, conducted by people on people and not on objects as such, where each actor has his/her own reality that influences his/her computational thinking.

4.3.3 Axiology – The role of values and ethics

Axiology is the role the researcher's values play in the research. The interpretation is linked to the real-life experiences, which is embodied in a practical world of information technology and education. Having identified the assumptions based on the epistemological, ontological and axiological stance (section 4.3), these assumptions inform the research paradigm, which is discussed in the next section. This research is positioned as part of an interpretivist/pragmatist philosophy in order to create an IT artefact (Section 1.11), based on the researcher's doubts and beliefs (Saunders, Lewis & Thornhill, 2019).

4.4 Research approach

According to Saunders, Lewis and Thornhill (2019), there are several approaches to conduct research, which are through deduction, induction and abduction. Creswell (2014), on the other hand, recognises quantitative, qualitative and mixed methods as three approaches. Saunders et al. (2019) regard the three approaches of Creswell (2014) as strategies. This research followed the Saunders, Lewis and Thornhill (2019) approach. Nicholls (2009), Clarke and Braun (2013) and Woiceshyn and Daellenbach (2018) argue that qualitative research is congruent with an inductive approach, in developing theory as opposed to quantitative research, that aligns with a deductive approach. According to Woiceshyn and Daellenbach (2018), deduction entails moving from general to the specific and induction moving from the specific to general, which results in abduction when the research moves in both directions.

In this research, the researcher investigated the relationship between APOS theory, programming concepts and the realisation of the data generated through interventions put in place, and the induction of theory in general. The Greenfoot programming language (section 2.2.2) provides these programming concepts practiced by learners in such a manner that the

APOS theory's mental structures are realised through mental mechanisms (interiorisation, encapsulation, de-encapsulation, coordination, reversal, generalisation and thematisation) to foster computational thinking among learners.

4.5 Research strategy

The researcher explored the role of computational thinking among high school learners and how computational thinking could be promoted among high school learners. The research strategy is discussed next under the following headings: (i) a demonstration case; (ii) action research; (iii) DSR strategy; (iv) intervention development; (v) interviews and observations; (vi) data collection strategies; (vii) sampling; and (viii) data analysis.

4.5.1 The demonstration case

The discussion is subdivided into the background and school visiting/planning process.

4.5.1.1 Background

A private school in Durbanville, Western Cape was the case for this research. Attendance was kept as illustrated in [Figure 4.4](#). The reason behind choosing the private school was the high standard of education at this private school in that they also taught robotics at lower grades and learners used tablets to conduct their everyday school activities. Robotics and tablets were unknown to public schools when conducting this research. Public schools now venture into robotics and tablet learning where resources are available, which makes the research case more relevant when teachers want to apply the outcomes of this research at a public school. A further motivation for choosing a private school is that teachers of one public school were hesitant to participate using an LMS, as it brought negative perceptions of work being added to their existing workload. During an information meeting with teachers of a public school, the researcher found that these teachers are not at all familiar with an LMS or the rollout of e-Learning in general. The vice-principal acknowledged that they are investigating the implementation of e-Learning within the school.

Using the private school as case may have introduced some bias, as the assumption is that the learners already have some computer skills in e.g. elementary programming or coding. The benefit of deliberately choosing this case lies with the assumption that the school already has the infrastructure required for the research and that the learners already obtained some level computer literacy.

4.5.1.2 The school visiting and planning process

[Figure 4.4](#) is as an example of a three-month visiting schedule to the private school in 2014 and 2015. The IT teacher signed off each visit, which formed part of the learner's daily

routine and the researcher's quality control. The visits did not enforce a different routine on the learner other than their learning content being more IT-programming language specific.

LOG ENTRIES FOR RESEARCH FOR W.C. ROTHMAN										CONFIRMATIO	SIGNATURE
NO	DATE	DISTAN	DEPART	GROUP	LOCATION	CLASS ST	CLASS EP	ACTIVITY	TEACHER	V/D Vyver	
1	16-07-2014	14	PROTEA VALLEY	8.3	WED	CURRO PRIVATE	11:47	12:35	Intro & Vid#1		
2	21-07-2014	28	CAPE TOWN	8.1	MON	CURRO PRIVATE	12:55	13:42	Intro & Vid#1		
3	22-07-2014	14	PROTEA VALLEY	8.1	TUE	CURRO PRIVATE	08:20	09:06	Intro & Vid#1		
4	23-07-2014	14	PROTEA VALLEY	8.3	WED	CURRO PRIVATE	08:20	09:06	Intro & Vid#1		
5	25-07-2014	28	CAPE TOWN	8.1	FRI	CURRO PRIVATE	11:00	11:47	LESSON 1		
6	28-07-2014	28	CAPE TOWN	8.3	MON	CURRO PRIVATE	12:55	13:42	LESSON 1		
7	30-07-2014	28	CAPE TOWN	8.3	WED	CURRO PRIVATE	11:47	12:35	LESSON 1		
8	04-08-2014	28	CAPE TOWN	8.1	MON	CURRO PRIVATE	12:55	13:42	LESSON 1		
9	05-08-2014	14	PROTEA VALLEY	8.1	TUE	CURRO PRIVATE	08:20	09:06	LESSON 2		
10	06-08-2014	14	PROTEA VALLEY	8.3	WED	CURRO PRIVATE	08:20	09:06	LESSON 2		
11	08-08-2014	28	CAPE TOWN	8.1	FRI	CURRO PRIVATE	11:00	11:47	LESSON 2		
12	11-08-2014	28	CAPE TOWN	8.3	MON	CURRO PRIVATE	12:55	13:42	LESSON 2		
13	13-08-2014	28	CAPE TOWN	8.3	WED	CURRO PRIVATE	11:47	12:35	LESSON 3		
14	18-08-2014	28	CAPE TOWN	8.1	MON	CURRO PRIVATE	12:55	13:42	LESSON 3		
15	19-08-2014	14	PROTEA VALLEY	8.1	TUE	CURRO PRIVATE	09:15	09:54	LESSON 3		
16	20-08-2014	14	PROTEA VALLEY	8.3	WED	CURRO PRIVATE	08:20	09:06	LESSON 3		
17	22-08-2014	28	CAPE TOWN	8.1	FRI	CURRO PRIVATE	11:23	12:00	LESSON 4		
18	25-08-2014	28	CAPE TOWN	8.3	MON	CURRO PRIVATE	12:40	13:35	LESSON 4		
19	27-08-2014	28	CAPE TOWN	8.3	WED	CURRO PRIVATE	11:47	12:35	LESSON 4		
20	01-09-2014	28	CAPE TOWN	8.1	MON	CURRO PRIVATE	12:40	13:35	LESSON 4		
21	02-09-2014	14	PROTEA VALLEY	8.1	TUE	CURRO PRIVATE	09:15	09:54	LESSON 5		
22	03-09-2014	14	PROTEA VALLEY	8.3	WED	CURRO PRIVATE	08:20	09:06	LESSON 5		
23	05-09-2014	28	CAPE TOWN	8.1	FRI	CURRO PRIVATE	11:23	12:00	LESSON 5		
24	08-09-2014	28	CAPE TOWN	8.3	MON	CURRO PRIVATE	12:55	13:42	LESSON 5		
25	10-09-2014	28	CAPE TOWN	8.3	WED	CURRO PRIVATE	11:47	12:35	TEST 1		
26	15-09-2014	28	CAPE TOWN	8.1	MON	CURRO PRIVATE	12:40	13:35	TEST 1		
27	16-09-2014	14	PROTEA VALLEY	8.1	TUE	CURRO PRIVATE	08:20	09:06	LESSON 6		
28	17-09-2014	14	PROTEA VALLEY	8.3	WED	CURRO PRIVATE	08:20	09:06	LESSON 6		
29	19-09-2014	28	CAPE TOWN	8.1	FRI	CURRO PRIVATE	11:23	12:00	LESSON 6		

Figure 4.4: One term's visits to the private school

These classes were managed by the researcher in his capacity as teacher and researcher, identified or selected by the IT teacher, which coincided with the researcher's schedule availability. The timetable also stretched over two weeks and Figure 4.5 and Figure 4.6 show classes in green on a two-week period timetable.

	09-Feb	10-Feb	11-Feb	12-Feb	13-Feb	16-Feb	17-Feb	18-Feb	19-Feb	20-Feb
	Ma	Di	Wo	Do	Vry	Ma	Di	Wo	Do	Vry
	Dag 1	Dag 2	Dag 3	Dag 4	Dag 5	Dag 6	Dag 7	Dag 8	Dag 9	Dag 10
Per 1:08h20-09h06	Saal				08h30		08h30			08h30
Per 2:09h15-09h54										
Per 3	10h40			10h40	10h25		10h25		10h40	10h25
Per 4										
Per 5:11h30-12h10					9.1				12h35	
Per 6:12h10-12h50	12h35			12h35				9.3	9.3	
Per 7:13h10-13h50			9.3							

Moet 11h45 ry

	23-Feb	24-Feb	25-Feb	26-Feb	27-Feb	02-Mar	03-Mar	04-Mar	05-Mar	06-Mar
	Ma	Di	Wo	Do	Vry	Ma	Di	Wo	Do	Vry
	Dag 1	Dag 2	Dag 3	Dag 4	Dag 5	Dag 6	Dag 7	Dag 8	Dag 9	Dag 10
Per 1:08h20-09h06	Saal	9.1								
Per 2:09h15-09h54			9.1							
Per 3										
Per 4			10h15	10h40				10h15	0h40-12h35	
Per 5:11h22-11h58					9.1			12h05	9.3 11h47 12h35	
Per 6:12h11-12h50			12h05	12h35				9.3		
Per 7:13h10-13h50			9.3							

Moet 11h20 ry

Moet 11h50 ry

Figure 4.5: Time table structured on a two-week period for first term

	09-Mar	10-Mar	11-Mar	12-Mar	13-Mar	16-Mar	17-Mar	18-Mar	19-Mar	20-Mar	
	Ma	Di	Wo	Do	Vry	Ma	Di	Wo	Do	Vry	Moet 11h45 ry
	Dag 1	Dag 2	Dag 3	Dag 4	Dag 5	Dag 6	Dag 7	Dag 8	Dag 9	Dag 10	
Per 1:08h20-09h06	Saal	9.1			08h30						
Per 2:09h15-09h54			9.1								
Per 3		10h40	10h15		10h25			10h15	10h40		
Per 4											
Per 5:11h22-11h58					9.1			12h05	12h35		Moet 11h45 ry
Per 6:12h11-12h50		12h35	12h05					9.3	9.3		
Per 7:13h10-13h50			9.3								

	23-Mar	24-Mar	25-Mar	26-Mar	27-Mar	30-Mar	31-Mar	01-Apr	02-Apr	03-Apr
	Ma	Di	Wo	Do	Vry	Ma	Di	Wo	Do	Vry
	Dag 1	Dag 2	Dag 3	Dag 4	Dag 5	Dag 6	Dag 7	Dag 8	Dag 9	Dag 10
Per 1:08h20-09h06	Saal	9.1	9.1							
Per 2:09h15-09h54										
Per 3										
Per 4										
Per 5:11h22-11h58										
Per 6:12h55-13h42			9.3							
Per 7				AAND						

Figure 4.6: Time table structured on a two-week period for second term

4.5.2 Action research (AR)

From the outset of the research, AR was considered the strategy of choice. Rapoport (1970:499) sees AR as contributing “both to practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within [a] mutually acceptable ethical framework”. According to Hult and Lennung (1980), AR assists in practical problem-solving, it expands scientific knowledge, and it enhances the competencies of the respective actors simultaneously. The authors further argue that AR increases understanding of a given social situation if performed collaboratively in an immediate situation, using data feedback in a cyclical process. Susman and Evered (1978), Page and Meyer (2000), Coghlan and Brannick (2001) and Järvinen (2007) regard AR as a set of steps, depicted in Figure 4.7, namely, diagnosing, action planning, action taking, evaluating and specifying learning.

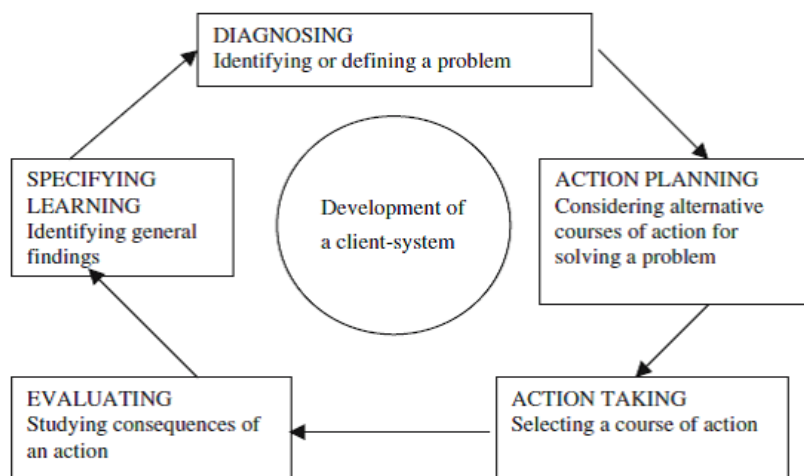


Figure 4.7: Five phases of Action Research Method (Adopted from Susman & Evered, 1978 as illustrated by Järvinen, 2007:39)

Page and (Meyer 2000) also propose an elaborate AR model with the following phases: (i) diagnosis; (ii) data collection; (iii) feedback and participant work; (iv) action planning; (v) action implementation; and (vi) evaluation. Keegan (2016) proposes a similar framework but adds interactive reflection as part of the stages.

This spiral process of reflection and action is rigorously pursued in all AR projects. Three main benefits of using AR are: (i) the contribution to the professional development of the participant, in this case the teacher; (ii) its capacity to generate knowledge and new practices; and (iii) the value of the teacher or researcher being part of the research. Furthermore, motivational aspects to use AR as research strategy are: (i) "It is a method of doing case study research" (Gummesson, 2000:83); (ii) It is a "type of applied research, designed to find an effective way of bringing about a conscious change in a partly controlled environment" (Collis & Hussey, 2014:67); and (iii) the research can be conducted within a single organisation (Coghlan & Brannick, 2001).

AR was chosen because, according to Hill and Scott (2004), AR results in richer data streams and provides deeper insights into the data by using a qualitative or quantitative approach. Furthermore, the choice was made because of all the above-mentioned reasons being congruent with the researcher's ontological and axiological stance. The initial research was built on identifying the problem of why learners did not perform in mathematics. The approach was to intervene and discover the reason for non-performance, which led to computational thinking being the motivation for the research. This led to investigating the impact of a programming language on the development of computational thinking among learners through a number of interventions, which affected mathematical learning. The result could have been merely a "yes-no" answer to determine if the mathematics marks improved or not after a programming language was introduced. However, for this study the result would have been insufficient, as the "yes-no" answer lacks a deeper insight into the problem of computational thinking development, at a level of formal operations necessary for proper learning to take place in Mathematics and Science. More in-depth analysis was thus required to fulfil the aim of the study.

As the study progressed, it became clear that AR did not suit the aim of the study. Although AR, according to Baskerville (2008), produces an artefact, social or organisational change is not part of the outcome. According to the FEDS framework (Venable, Pries-Heje & Baskerville, 2016), a Human Risk and Effectiveness strategy was used that focused on formative and naturalistic evaluation. This was done during reflection on the instance domain (Gregor, Müller & Seidel, 2013), which showed that AR was not the suitable choice, as the outcomes did not fully match the expectations. A richer depth of data was needed at an educational level to bringing about change, especially because a wicked problem (section

3.3) was investigated. The researcher then decided to look at Design Science (DS) as strategy. The DS paradigm originates from engineering and artificial science, as advocated by Simon (1996).

DS focuses on artificial objects and phenomena that serve human purposes. Hevner et al. (2004) introduced the theory behind DS research for information systems. The solution produced by DS research is an artefact, which may take on the form of a construct, model, method or instantiation (March & Smith, 1995). DS for this research was then executed by the researcher as Design Research (DR), as Winter (2008) suggests. A generic solution is developed and supported by the theory behind the artefact. Figure 4.8 shows a comparative study by van Wyk and de Villiers (2018) of DR, DSR and DBR.

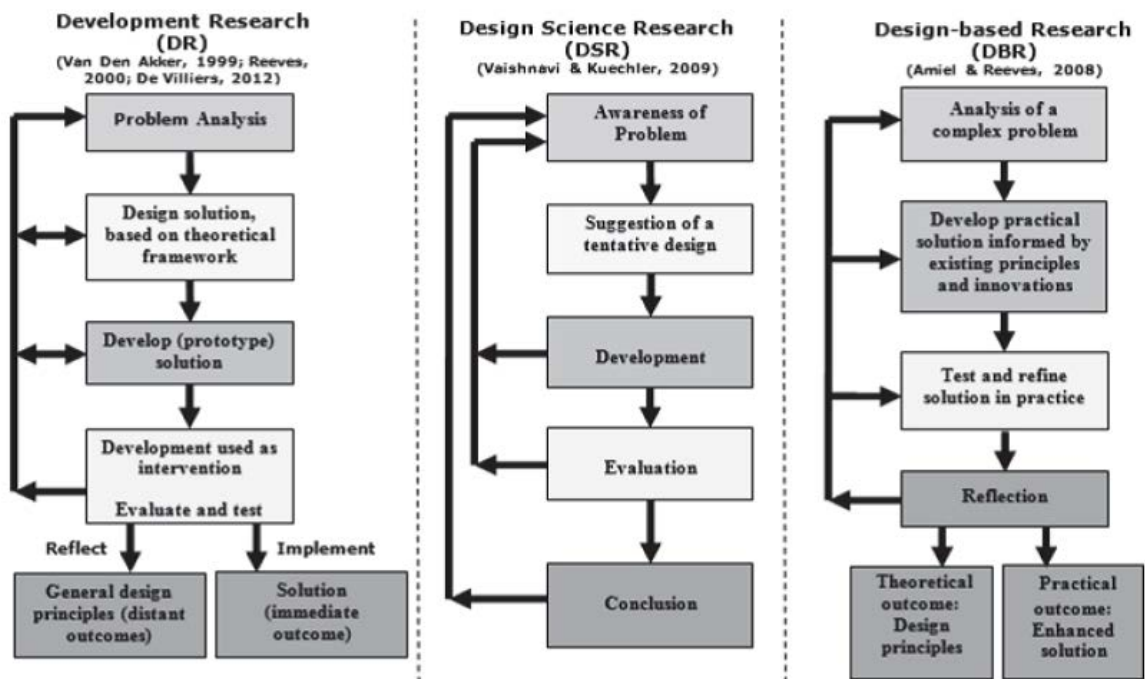


Figure 4.8: Composite representation of DR, DSR and DBR (Adopted from Van Wyk & De Villiers, 2018:304)

4.5.3 Design Research (DR) strategy

An in-depth discussion on DR as the strategy of choice for this research is now discussed. Figure 4.9 represents a flow of the discussion.

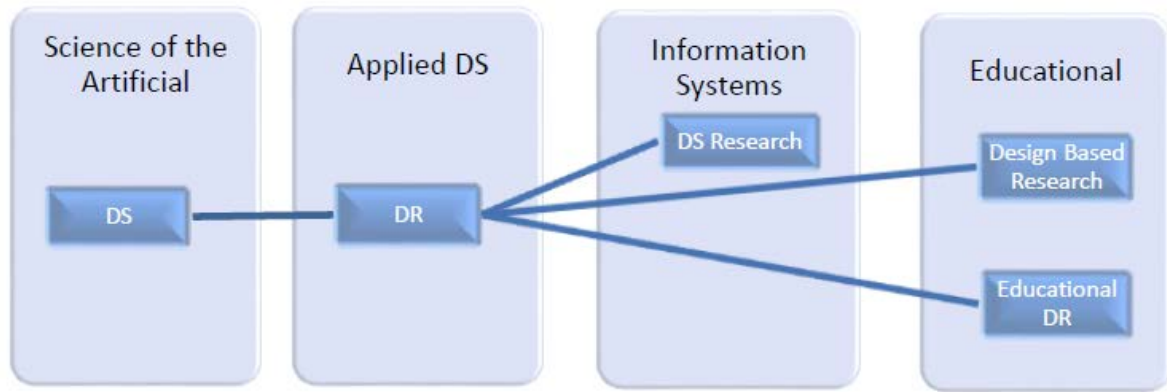


Figure 4.9: Relationship among DS, DR, DSR and EDR (Adapted from Venable, Pries-Heje & Baskerville, 2016:141; Miah, Solomonides & Gammack, 2010:2)

The next section focuses on the interventions that were designed and implemented in this research as part of the DR strategy. The interventions were developed and implemented according to the EDR model of van Wyk and de Villiers (2018), combined with the FEDS framework (Venable, Pries-Heje & Baskerville, 2016) to emphasise the evaluation phase and reflection phase (Gregor, Müller & Seidel, 2013).

4.5.4 Intervention development

4.5.4.1 Introduction

The interventions for this research were structured as an exploratory sequential design, a building method, followed by the interpretation and reporting. The formative evaluations of the interventions were repetitively criticised based on relevance, consistency, practicality and effectiveness, as depicted in [Table 5.4](#). Each intervention was grounded in themes that developed from the APOS intervention. Note that Interventions 5 and 11 consist of sub-interventions A and B. Interventions 2 and 3 consist of three sub-interventions, A, B and C. In the case of Intervention 2, further interventions were needed and marked as such. Intervention 14 has five sub-interventions. The balance of the interventions (4, 7, 8, 9, 10, 11 & 12) consists of one intervention. In the cases where there were sub-interventions, the interventions were numbered with A, B, C, etc. as a post-fix.

4.5.4.2 Intervention 1: Abstraction (abstract thinking) assessment (Appendix B-1)

Intervention 1 was a starting point to determine if abstraction was absent, which then called for action from the researcher. The measuring of abstraction can be done in different ways, and the one in [Figure 4.10](#) was chosen as a fun exercise for learners. The important aspect was to create a “buzz” among learners and that they enjoyed the challenge. The intervention, as in prior discussions, consists of three sections: Design, Method and Interpretation.

(a) Design


Exploratory Sequential Design (ESD) was used for this research and started with exploring, collecting and analysing data from a qualitative perspective through observation and a test in which the user had to indicate if s/he is 'on the floor' or 'on top of step', after taking 239 steps. This informed quantitative data collection in sequence. [Figure 4.10](#) shows the first intervention, which informed further interventions or iterations thereof, investigating computational thinking. Abstraction and automation are the pillars of computational thinking, where abstraction is the inherent quality of any learner or person. Hence, the focus was on abstraction and not automation.

(i) Qualitative methodology of Intervention 1

During this intervention, the abstraction skills of learners were explored. This was done because computational thinking consists of abstraction and automation (Wing 2008, 2011; Bocconi et al., 2016; Cetin & Dubinsky, 2017). The exercise ([Figure 4.10](#)) was handed out to learners, and through observation, the interactions of the learners were studied. In this handout, learners had to determine the answer to the problem by stating if the figurine is on top of the step or on the ground after 239 steps. In [Figure 4.10](#), the figurine starts out by standing in front of the bench with both feet on the floor. The researcher or teacher can also give a dramatic illustration by stepping up and down using a chair.

Abstraction: Please make an X in the block of your choice.

After the person stepped up and down for 239 times, will he/she be On the Floor, on Top of the step, or is your answer a GUESS



ON THE FLOOR	ON TOP OF STEP	MUST GUESS

Figure 4.10: Abstraction exercise

The time limit on the handout was set to one minute and the researcher used a stopwatch. Each learner had to operate individually and the researcher used monitors in the class who prevented learners from communicating with one another in order to obtain the answer. After one minute elapsed, the learners indicated their answers with a cross (X) in the designated

block and all the forms were placed in a collection box. The cross had to be drawn where learners determined the figurine would stop, either on the floor (incorrect position) or on top of the step (correct position). No names were required on these forms so as to protect learners' identity and motivate learners to be truthful in their answers.


(ii) Quantitative methodology on Intervention 1

This intervention then produced quantitative data as illustrated by the container in [Table 4.1](#), summarising the abstraction skills of both groups of learners. The analysis was done using an Excel spreadsheet and implementing pivot tables and the data can be seen in [Table 5.6](#).

Table 4.1: Abstract thinking among grade 8 learners

Abstraction: Please make an X in the block of your choice.

After the person stepped up and down for 239 times, will he/she be On the Floor, on Top of the step, or is your answer a GUESS



ON THE FLOOR	ON TOP OF STEP	MUST GUESS

GROUPS	Finish on Floor	Finish on Step	Guesses -	TOTAL
A	-	-	-	-
B	-	-	-	-
TOTAL	-	-	-	-

Having the data at hand, an analysis should show that the current data collection procedures lead to the data collection of the next procedure, the latter building onto the former (Fetters, Curry & Creswell, 2013).

(b) Method

The current intervention informs the researcher of the degree of abstraction that does exist. Intervention 2 ties in with the literature review and the EDR approach, where the motivation and implementation of Greenfoot as programming language is identified and rolled out ([Appendix C](#)).

(c) Interpretation

Using narrative as integration procedure, the qualitative and qualitative findings were described by means of a single or series of reports. Each intervention was grounded in

themes that developed from the intervention Action, Process, Object and Schema, which are mental structures that inform mental mechanisms – interiorisation, coordination, reversal, encapsulation and thematisation – and which promote computational thinking.

4.5.4.3 Intervention 2: Implement Greenfoot programming language (Appendix C)

The researcher or teacher established that abstraction in its practical form was a challenge. The learners’ thoughts on mathematics and mathematical concept images governed mathematical concept definitions. The researcher employed a programming language to investigate computational thinking through a programming language aligned to APOS as stated in the “how” questions of RQ2. SRQ 2.1 reads: “How are the constructs of a programming language taught among high school learners at a cognitive level of formal operations?”

(a) Design

As stated earlier, Exploratory Sequential Design (ESD) commenced by exploring, collecting and analysing data from a qualitative perspective, which then informed qualitative data collection in sequence. Figure 4.11 shows computational thinking using a programming language as computational notation. Computational thinking was triggered through computation as the process employed when creating computational models or abstractions of a problem given to learners to solve.

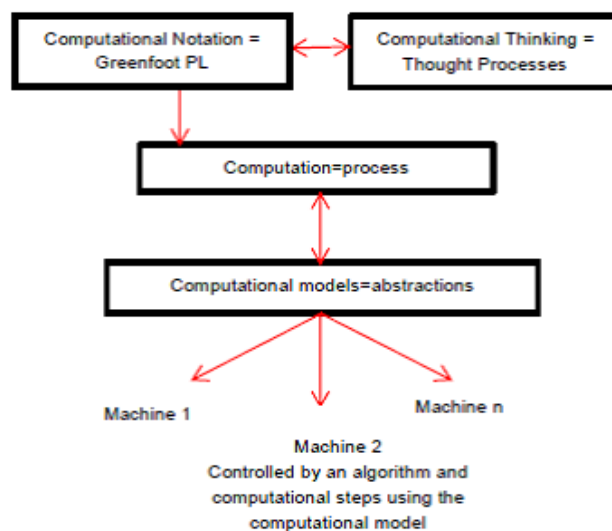


Figure 4.11: Computational thinking in motion (Adapted from Denning, 2017; Aho, 2011)

The intervention suggested that learners need to understand and know the computational notation selected for this research. This led to other challenges such as knowledge of the programming environment, which could not simply be accepted as a fact.

(b) Method

The intervention triggered a bevy of pre-requisites that were needed for learners in order to roll out a programming language effectively in order to achieve computational thinking, unless the learners were knowledgeable in a programming language. The methodology did not even consider APOS theory at this stage, as the Greenfoot programming language was unknown to learners. Learners did not understand the interface of a programming language or how to work with the interface to launch a simple program. The trigger of this intervention created a roll-back to the pre-requisites the learner needed in order to make progress, which initiated intervention 2A.

(c) Interpretation

The interpretation and reporting were done through observation. What was significant at this stage of the process was that the researcher was locked into pre-requisites needed by learners to become fluent in a programming language, as discussed in section 5.4.3.3(b).

4.5.4.4 Intervention 2A: Introduction of a Genetic Decomposition process (Adapted from Arnon et al., 2014:112; Appendices D-1, D-2)

Intervention 2A was a starting point to understand how to roll out the concept of activities, classroom discussions and exercises (**ACE**) (Dubinsky, 1991; Arnon et al., 2014). The researcher was guided by this framework to ensure success. Figure 2.20 portrayed in Appendix D-1 was composed after the literature study in section 2.2.2.2(b)(v).

(a) Design

ESD started with exploring, collecting and analysing the data from a qualitative perspective. The rollout was based on ACE and the rollout process was called genetic decomposition (Arnon et al., 2014), as it should change the way the learners approached and looked at the problem of using the IDE to accomplish programming. The outcome created a schema which should form part of the learners' concept definition of a programming language.

(b) Method

The proposed schema is created by following the steps in Figure 2.20. The intervention 2B is triggered based on the framework.

(c) Interpretation

EDR is based on an intervention framework and hence the results of each step are reported using a multi-stage methods approach. These multi-stages can contribute to satisfying the research questions.

4.5.4.5 Intervention 2B: Introduction of an enhanced Genetic Decomposition of “Load a Greenfoot Scenario” (Appendix D-2)

Intervention 2B is a specific genetic decomposition that was created from observing the learners based on the GD framework, as reported in [Appendix D-1](#).

(a) Design

ESD was followed to explore how learners engaged with the IDE and a sequence of steps was emerged from these observations to let learners accomplish the inner workings of the Greenfoot IDE. During this intervention, the abstraction skills of learners were explored.

(b) Method

The intervention 2B is rolled out where the “Help” documentation was explored to allow learners to be autonomous when exploring the Greenfoot programming language.

(c) Interpretation

The specific genetic decomposition was broken down as processes and refinements. Each process described as 1P, where the “1” numeral indicated the ordinal position of the process, was aligned with the “R” or refinements indicated as 1.1 and 1.2 and so on. The refinements described each process in detail to indicate what was expected of the learner for the researcher to have a specific rollout plan of action.

4.5.4.6 Intervention 2C: Help documentation in Greenfoot (Appendix D-3)

Intervention 2C was a starting point to understand the need among the designated group(s) of learners, how the “Help” menus may assist learners in their quest to become self-autonomous. A typical constructionist approach was followed, built on a truth basis as put forward by experts in the field.

(a) Design

The learners were given exercises to explore the “Help” menus in order to better understand the Greenfoot programming language. The Greenfoot programming language consists of different classes grouped in libraries. The learners were required understanding the use of these structures in order to solve problems and develop algorithms.

(b) Method

Conduct classes using flipped classroom techniques and assess what the learners understand through a questionnaire. However, cognitive load theory dictates that the extraneous cognitive load must be minimised. The next procedure that followed was

Intervention 3A, triggered to ensure a trusted source of information and mitigate the relative epistemological dilemma caused by constructivist approaches.

(c) Interpretation

The learners were observed and assessments (Appendix E-3) were given to them to interpret their understanding of the Greenfoot family of classes.

4.5.4.7 Intervention 3: Interaction with the Moodle LMS (Appendix E-1)

Circumscription was applied because of the complex nature of introducing Moodle without a strategy. The strategy is discussed through sub-interventions 3A, 3B and 3C.

4.5.4.8 Sub-Intervention 3A: Introduction of the Moodle LMS (Appendix E-1)

Sub-Intervention 3A focused on creating an LMS for learners as a credible source of information to help them when conducting research on Greenfoot programming language.

(a) Design

Learners were given a list of prerequisites (Appendix E-1), which informed them of how to approach the Moodle LMS and what pre-requisites were allowed in class.

(b) Method

The list of prerequisites is supposed to motivate learners to participate with more enthusiasm as it allows them the use of their cell phones and headphones, and watching videos. Intervention 3B is triggered, and it entails a practical approach to make the LMS approach a reality for each learner.

(c) Interpretation

The researcher had to determine whether the list of pre-requisites was acceptable to these grade 8 learners and how the list was received.

4.5.4.9 Intervention 3B: Juggling enactment to enforce Moodle usage among learners (Appendix E-2)

Intervention 3B was a fun activity to draw learners into using Moodle LMS as resource and repository in order to guide them on how to develop the skill of juggling.

(a) Design

Each learner was given the opportunity to enact the process of juggling. It is a difficult process if the learner approaches this activity without the basic mental structures of the APOS theory. The researcher relied on observation and taking notes. Videography was done to capture the learners' phases of development.

The link is:

https://teams.microsoft.com/_#/school/files/General?threadId=19%3A8eff649f6647440ea80e6841eb4dcce8%40thead.tacv2&ctx=channel&context=NavorsingsVideos%2520Curro%2520WCED&rootfolder=%252Fsites%252FResearch742%252FShared%2520Documents%252FGeneral%252FNavorsingsVideos%2520Curro%2520WCED
(Currently on TEAMS).

(b) Method

The learners were given a questionnaire (Appendix E-2) on which they reported their progress and reflected on their experience as to mould APOS theory and the exercise together. The process led to Intervention (3C), with the purpose of determining whether an LMS (in the case the Moodle LMS) is an important element towards learning.

(c) Interpretation

The narrative as integration procedure was used to describe the qualitative findings by means of a single or series of reports.

4.5.4.10 Intervention 3C: Moodle and generalised terminology (Appendix 3C)

The aim of Intervention 3C is to verify if Moodle is a reliable source of information for the learners.

(a) Design

ESD started with exploring, collecting data from the learners, and analysing the data by providing a questionnaire to determine the knowledge and involvement of the learners.

(b) Method

The questionnaire (Appendix E-3) involved each learner to reflect on their involvement with Moodle LMS and the Greenfoot programming language. The outcome led to intervention 4A, which was triggered by providing learners with access to Moodle, considering cost of ownership and ease of configuration.

(c) Interpretation

The Moodle LMS provided “flipped classroom” effects in the form of videos and forums for these learners to participate in. The outcome informed intervention 4A as depicted in Appendix F-1, which necessitated a Moodle LMS.

4.5.4.11 Intervention 4: Creating a Moodle Learner Management System (LMS)

Because of the complex nature of rolling out a Moodle LMS, this intervention was circumscribed. The complex intervention is rolled out as interventions 4A and 4B.

4.5.4.12 Intervention 4A: Creating a Linux Server with external access (Appendix F-1)

Intervention 4A was initiated to develop an in-house Linux server to accommodate learners in a school with poor IT resources, but still provided state of the art access to resources for teaching and learning.

(a) Design

A Linux server was built from scratch using a throwaway PC at the school. The Moodle LMS as resource was constructed and rolled out. Internet connections were tested and implemented to allow global access. The functionality had to be tested in circumstances where larger populations were replicated and the impact on the Linux server was assessed.

(b) Method

The Linux server acted as a repository and informed learners as a knowledgeable resource. However, many challenges emerged that led to Intervention 4B and which were triggered to mitigate any possible challenges from Intervention 4A.

(c) Interpretation

The researcher needed to record all challenges and investigate how these challenges could be overcome to promote teaching and learning for the learners and the teacher who managed the initiative.

4.5.4.13 Intervention 4B: Creating a cloud-based Moodle LMS (Appendix F-2)

Intervention 4B was a natural decision made to overcome any challenges caused by the previous intervention (4A) to assist learners in teaching and learning.

(a) Design

Exploring all possibilities, a web-based initiative was designed, focusing on the actual aim of the initiative. The content remained the same, but the idea was to focus on the actual aim of the initiative and not on connectivity as an obstacle.

(b) Method

Intervention 4B provided the necessary connectivity for teaching and learning to take place. This triggered the original goal of the study, i.e., teaching the Greenfoot programming language to develop abstraction and computational thinking among learners. The next intervention (5A) was triggered, which entailed the rollout of the Greenfoot programming language. This intervention entailed that learners' focus is on Greenfoot and not on the 'sideshows' of programming language usage.

(c) Interpretation

Following an internet-based approach, the researcher applied Moodle functions and developed more professional usage of all Moodle LMS capabilities rather than focusing on enhancing connectivity and maintaining the Linux server. The focus is on the instructional design of Greenfoot using the Moodle LMS to support cognitive load theory.

4.5.4.14 Intervention 5: Greenfoot access

This intervention was circumscribed due to learners being lost within the IDE of Greenfoot programming language; the researcher needed to backtrack to deal with this gap in the learners' understanding. The intervention is rolled out as interventions 5A and 5B.

4.5.4.15 Intervention 5A: Introduction to Greenfoot (Appendix G-1)

Intervention 5A was a starting point to explore and understand how the Greenfoot programming language affects learners and their teaching and learning activities in line with this research.

(a) Design

The Greenfoot programming language was implemented by following a constructionist approach. The link to the videos on the development of scenarios was made available on the Moodle LMS for learners to investigate prior to algorithm implementation. The link is:

https://teams.microsoft.com/_#/school/files/General?threadId=19%3A8eff649f6647440ea80e6841eb4dcce8%40th read.tacv2&ctx=channel&context=NavorsingsVideos%2520Curro%2520WCED&rootfolder=%252Fsites%252FResearch742%252FShared%2520Documents%252FGeneral%252FNavorsingsVideos%2520Curro%2520WCED

(on MS TEAMS).

(b) Method

The purpose of intervention 5A was to expose and introduce the Greenfoot programming language to the students, which triggered intervention 5B, to revisit previous mathematical concept definitions and restore the original perspective of APOS theory for this initiative.

(c) Interpretation

Using narrative as integration procedure, the qualitative findings were described by means of a single or series of reports.

4.5.4.16 Intervention 5B: Revisit previous activities (Appendix G-2)

Intervention 5B was a revisit of confirming the aim of this research to keep the focus on APOS theory and mathematics. The learners had to maintain their perspective on the mathematical thinking processes.

(a) Design

ESD was done to understand how the learners relate to programming, mathematics and APOS theory.

(b) Method

The learners were given a questionnaire (Appendix G-2) based on their activities done in class. The next intervention (6) was triggered, which illustrated APOS theory based on a mathematical concept definition of simplification. This was triggered to apply the outcome of the previous intervention (5B).

(c) Interpretation

Through observation and videos, the outcomes were recorded for direction on whether additional help was still needed to assist learners with the APOS concept.

4.5.4.17 Intervention 6: Applying Process and Object within mathematics (Appendix H)

Intervention 6 confirmed APOS theory as a starting point for the researcher to understand the need among the designated group(s) of learners, to determine whether abstraction is absent and calls for action from the researcher.

(a) Design

ESD was used to explore and understand how mathematical concept definitions interact with APOS theory. The learners were given a worksheet (Appendix H) which they had to complete to reflect on APOS theory in order to determine how they relate to mathematical problem solving.

(b) Method

The questionnaire handed out to learners contained questions on simplification and the APOS theory as underlying thinking strategy. The outcome of this reflection triggered intervention 7, where the mental structures are developed within the Greenfoot programming language.

(c) Interpretation

Learners need to link APOS theory within mathematics. The reflection guide learners to visualise the APOS theory mental structures and mechanisms from a Greenfoot perspective, which triggered the next intervention. This might relate to the previous intervention that used mathematics concept definitions as the sole example.

4.5.4.18 Intervention 7: Greenfoot as Process and Object (Appendix I)

Intervention 7 was a complete rollout of Greenfoot to let learners experience APOS theory when implementing an algorithm in the Greenfoot programming language.

(a) Design

ESD was used to implement a scenario in the Greenfoot programming language. The APOS theory mental structures and mechanisms are monitored through observation and the assessment of learners and their implementations.

(b) Method

The Greenfoot programming language scenario was used to reflect on their programming language skills with specific Greenfoot references. Intervention 8 was triggered and focused on the usage of Greenfoot as programming language and provided coding as part of the rollout.

(c) Interpretation

Using narrative as integration procedure, the qualitative findings are described by means of a single or series of reports. The learners were then assessed to produce code rather than clicking on menus, which illustrated the dual modality of Greenfoot. Interpretation is one of the important components of computational thinking in that the development of computational thinking requires language.

4.5.4.19 Intervention 8: Rollout of code in Greenfoot (Appendix J)

Intervention 8 was rolled out to let learners experiment and discover coding as the next level of developing computational thinking ([Figure 5.35](#)).

(a) Design

ESD was used to explore and understand how coding influenced computational thinking among learners. A constructionist approach was followed to let learners investigate an algorithm and bring it into perspective of using Greenfoot as computational notation and developing computations.

(b) Method

The learners were given a task sheet (Appendix J) to create an algorithm for a problem. This led to several lines of coding, which triggered Intervention 9 as a natural development to encapsulate all these lines of coding, thereby giving meaning to these code snippets through abstraction. The exercise was developed to enable mental mechanisms within the mental structure of the learner.

(c) Interpretation

Using narrative as integration procedure, the qualitative findings are described by means of a single or series of reports.

4.5.4.20 Intervention 9: Making decisions towards Encapsulation (Appendix K)

Intervention 9 was a logical development that entailed encapsulation of code.

(a) Design

ESD was used to explore and understand how encapsulation affected learners' thinking patterns in terms of the APOS Theory when considering Processes and Objects. The word Object has a duality when referring to APOS theory and the Greenfoot programming language.

(b) Method

Many lines of code led to abstraction of the code through nesting the code into a method. Learners found these abstraction methods helpful, which triggered Intervention 10. The encapsulation process was expanded to provide more meaning to learners in their understanding of encapsulation.

(c) Interpretation

Learners produce multiple lines of coding, which they investigate through applying mental mechanisms within a mental structure. The intervention was grounded in themes, which developed from the Action, Process, Object and Schema (APOS) theory, which are mental structures that inform mental mechanisms – interiorisation, coordination, reversal, encapsulation and thematisation – thereby promoting computational thinking.

4.5.4.21 Intervention 10: Revisit encapsulation with Randomize option (Appendix L)

Intervention 10 was a logical development stemming from Intervention 9, entailing the built-in function (Randomize), which is encapsulation of code.

(a) Design

ESD was used to explore and understand how Randomization would influence encapsulation in terms of the APOS theory when looking at Processes and Objects.

(b) Method

Randomization was used, which caused a natural flow of events. Learners were given the opportunity to be assessed in Intervention 11A. Learners were informed of assessment structures to motivate them and improve their results.

(c) Interpretation

Using narrative as integration procedure, the qualitative findings are described by means of a single or series of reports. The learners can now develop a game using random parameters.

4.5.4.22 Intervention 11: Assessment (Appendix M)

Assessment is another complex task and cannot simply be 'dumped' onto these learners without proper preparation; hence, the assessment was circumscribed (11A and 11B).

4.5.4.23 Intervention 11A: Informing learners of the assessment in a structured manner (Appendix M-1)

Intervention 11A was applied to compel learners to prepare for an assessment on Greenfoot programming language structures. The research did not form part of the traditional formal curriculum; there was a need to provide space for learners to enjoy an assessment and at the same time making the Greenfoot teaching and learning their own, with the focus on cognitive load theory.

(a) Design

ESD was used to explore and understand how assessments could prepare learners in terms of the APOS Theory in a positive manner.

(b) Method

Learners were taught with the focus on the assessment procedure. Learners had a specific guideline on how to prepare for assessment. Intervention 11B was the outcome where learners were assessed.

(c) Interpretation

Learners were taught with the focus on the assessment in order to at least make an effort to own the assessment procedure and prepare for these assessments directly. The concept definitions were highlighted in the learners' assessment preparation.

4.5.4.24 Intervention 11B: Assessment in Greenfoot on Encapsulation and problem solving (Appendix M-2)

Intervention 11B was the assessment. Learners could opt for an open book approach to find a solution to the problem. In this research, the assessment was a combination of Microsoft Paint to develop the background of the scenario, group work to overcome the slowness of the remote desktop (RDP) environment, and teamwork, completing the project in record time.

(a) Design

ESD was used to explore and understand how learners dealt with the assessment. The researcher incorporated video, observation and a questionnaire to collect data for the assessment rollout. The task at hand consisted of creating a total customised Greenfoot scenario, from the world as background to images and sound clips. All these components were stored in a specific folder and the learners had to understand the folder structure of Greenfoot to accomplish the project.

(b) Method

Learners used Microsoft Paint and sound clips which they constructed themselves. The folder structure was paramount to each learner's understanding. Although the assessment focused on basic programming, Intervention 12 was triggered when the learners recognised the need to use a variable in the Greenfoot programming language.

(c) Interpretation

Learners were taught with the focus on the assessment procedure, in order to make an effort to own the assessment procedures and prepare for these assessments specifically. With the current academic load, the researcher wanted to alleviate extraneous load as well. This intervention led to the assessment, where learners had to take ownership of the structures in the Greenfoot programming language. This led to Microsoft Paint creations and implementations of the folder structure, containing sound, images and code to provide the solution to the problem in the Greenfoot programming language, as set out in Appendix M-2. Every aspect of coding and structuring the project was associated with APOS theory, where learners had to answer on an individual basis their impressions and understanding of how APOS theory related to the project.

4.5.4.25 Intervention 12: The variable in Greenfoot (Appendix N)

Intervention 12 focused on introduction the variable into Greenfoot. A variable makes any programming language open-ended and problems can be solved using generalised solutions.

(a) Design

ESD was used to explore and understand how learners dealt with a variable in Greenfoot. The intervention was based on the manipulation of x and y axis values, linked to built-in methods. The Greenfoot problem (how to use variable), was interpreted to relate to the learners' current mathematical concept definition of graph theory. As mentioned previously, when replicating the research, the intervention must be based on the current mathematics concept definition explained in class at that time.

(b) Method

The method was built on using graph theory based on x and y values, which had to change value and hence was illustrated by moving an object through changing x and y values. This triggered Intervention 13, where the learners were given a problem to solve by applying the variable concept definition in Greenfoot.

(c) Interpretation

Learners were taught what a variable was and how a variable's values may change through x and y axis values. The example highlighted the Boolean and Integer variable. The scenario used was based on the rocket actor being moved either vertically or horizontally, which promoted complex learning.

4.5.4.26 Intervention 13: Moving from Process to Object in APOS using Greenfoot (Appendix O)

Intervention 13 focused on the introduction of a variable in Greenfoot.

(a) Design

ESD was used to explore and understand how learners dealt with a variable in Greenfoot. The intervention was based on the manipulation of x and y axis values, linked to the built-in methods `getX()` and `getY()`. The Greenfoot problem designed by the researcher using variables could be interpreted in relation to the learners' current mathematical concept definition of graph theory.

(b) Method

The learners were given a task that involved red and blue coloured balloons. These balloons were instantiations of a `balloon` actor class. The year came to an end regarding the research, and Intervention 14 was an obvious next intervention where learners were given a problem to solve in Greenfoot programming language, when they returned the next year.

(c) Interpretation

Learners were taught what a variable is and how a variable's values may change through changing x and y axis values. The example highlighted the Boolean and Integer variable. The scenario was changed for learners to make decisions through coding. The IF statement forms part of the decision structure in any language, and this statement was introduced to the learning using the Greenfoot programming language. The solutions to problems became more challenging, as the learners had to write general solutions to complex problems based on decisions.

4.5.4.27 Intervention 14: GD creation on IF statement

Conditional statements are complex structures, which need time to explain, especially to these learners acquiring programming language competencies.

4.5.4.28 Intervention 14A: Basic creation of scenario with World and Actor classes (Appendix P-1)

Intervention 14 focused on recapping the Greenfoot programming language. The learners were now in Grade 9 and the purpose of the intervention was to revisit the Greenfoot programming language.

(a) Design

ESD was used to explore and understand how learners experienced the Greenfoot programming language during the first year of the research. The exercise was a basic one that covered the understanding and application of the Greenfoot programming language scenario after the December holidays.

(b) Method

The learners were assessed through posing questions based on the problem they had to solve. Intervention 14B was triggered where learners associated the APOS theory mental mechanisms with the Greenfoot programming language structures.

(c) Interpretation

At the start of the New Year, learners were given a basic problem to solve through the Greenfoot programming language. A questionnaire (Appendix P-1) was given afterwards to determine if the students relate to APOS theory in terms of the mental structures and mechanisms.

4.5.4.29 Intervention 14B: Manipulation of Actors in a World (Appendix P-2)

Intervention 14B focused on exploring and understanding how the learners' thoughts were ordered using the Greenfoot programming language.

(a) Design

ESD was used to explore and understand how learners' thoughts were structured in the Greenfoot programming language. The exercise was a scenario containing all the necessary structures. The learners had to comment on their code and bring that into perspective through reflection.

(b) Method

Learners were given a questionnaire (Appendix P-2) to determine their knowledge of Greenfoot. Having the results at hand, intervention 14C was triggered, where learners had to create a solution using the Greenfoot programming language through built-in methods and control structures such as the IF statement.

(c) Interpretation

Learners were given a basic problem to solve through the Greenfoot programming language. Learners answered questions on a questionnaire to determine if they related to APOS theory mental structures and mechanisms. The questionnaire assessed if the learners could discuss the composition of a Greenfoot programming language scenario. This was to see if the learners thought about what they were doing through reflection and APOS theory.

4.5.4.30 Intervention 14C: Interaction of Actor within the World solving problems (IF statement as precursor to GD) (Appendix P-3)

Intervention 14C focused on exploring and understanding how the learners' thoughts were ordered using the Greenfoot programming language by placing the learner into a problem situation.

(a) Design

ESD was used to explore and understand how learners' thoughts were structured in the Greenfoot programming language in terms of making decisions using code. The exercise was a scenario containing all the necessary structures. The learners had to provide conditional code structures to solve a problem situation of the actor in the world.

(b) Method

Learners had to rely on the process component of APOS theory and relate to programming language objects and schemas. In the next intervention, 14D, the learners had to create a solution using the Greenfoot programming language using built-in methods and control structures such as the IF statement.

(c) Interpretation

Learners were given a basic problem to solve through the Greenfoot programming language. A questionnaire was given afterwards to determine if they related to APOS theory mental structures and mechanisms. They were given a questionnaire (Appendix P-3) to assess if they could discuss the composition of a Greenfoot programming language scenario. This was to see if the learners thought about what they were doing through reflection in APOS theory.

4.5.4.31 Intervention 14D: Adding graph paper as part of GD to develop algorithm (Appendix P-4)

Intervention 14D focused on exploring and understanding how the learners' thoughts were ordered using the Greenfoot programming language by creating a problem for learners to solve.

(a) Design

ESD was used to explore and understand how learners' thoughts were structured in the Greenfoot programming language, making decisions using code. Learners were given a problem and were compelled to go back to the Actions mental structure and pursue the problem with pencil and graph paper. The idea was to determine how many learners could visualise this problem in their minds without having to perform actions to solve the problem.

(b) Method

Learners were given a problem and they had to use graph paper and represent the scenario as a storyboard. This triggered intervention 14E, where the learners had to create a solution using the Greenfoot programming language and including the conditional IF structure to help them solve the problem represented on paper.

(c) Interpretation

Learners were given questionnaires (Appendix P-4) afterwards to determine if they related to APOS theory mental structures and mechanisms, developed a solution as a Process, and implemented decision structures using IF statements.

4.5.4.32 Intervention 14E: The IF statement as a solution to address problems (Appendix P-5)

Intervention 14E focused on exploring and understanding how the learners' thoughts were ordered using Greenfoot programming language, by creating a problem for learners to solve based on a condition.

(a) Design

ESD was used to explore and understand how learners' thoughts were structured in Greenfoot programming language making decisions using code. Learners were given a problem and were forced to go back to Actions mental structures and pursue the problem with pencil and graph paper. The idea was to detect through observation and then marking their answers, how many learners visualised this problem in their minds and not had to perform actions to solve the problem using code.

(b) Method

Learners were left to code what developed in the previous intervention 14D but using decision control structures in Greenfoot.

(c) Interpretation

The intervention assessed was grounded in themes namely actions, process, object and schema. These were highlighted in their task. Learners had to code what they drafted on paper. The researcher did not include the conclusion of these interventions by taking the interventions as a block-based rollout to teachers within the data. However, to fulfil the developmentalist paradigm in [Figure 3.4](#) teachers were given multiple opportunities to validate the researcher's actions through three weekend rollouts offered to three regions of schools in the Western Cape. This gave the researcher an opportunity to gauge the artefact from a more objective community. The emphasis thus shifted from interpretivist to a positivist paradigm.

4.5.4.33 Intervention 15: Testing Greenfoot to be accepted among teachers (Appendix Q)

Intervention 15 focused on exploring and understanding how the teachers in IT and CAT domains experienced the Greenfoot programming language. This intervention was condensed into a weekend block-based rollout.

(a) Design

ESD was used to explore and understand how teachers' thoughts were structured in terms of the Greenfoot programming language. Teachers were invited at three different occasions and in different regions of the WCED to attend the course on the Greenfoot programming language. The courses were organised by the WCED in conjunction with Oracle, and the researcher was the instructor.

(b) Method

The teachers were subjected to the same instructional methods covering the same work as the learners, but the time period was shortened. The intervention commenced on a Friday afternoon and continued throughout the Saturday.

(c) Interpretation

The assessed intervention was grounded in the action, process, object and schema themes.

4.5.4.34 Intervention 16: Creating an arcade game (Appendix U)

Intervention 16 was aimed at bringing together all Greenfoot programming language coding learnt and constructing a fully-fledged arcade game using graphics and code.

(a) Design

ESD was used to explore and understand how learners' thoughts were structured using Greenfoot programming language and applying APOS theory. Learners were exposed to a vast number of graphic designs and embedded mathematics.

(b) Method

The learners were exposed to videos on the Moodle LMS to show how the game was constructed using different components to form a unit. It entailed sound clips as well.

(c) Interpretation

The assessed intervention was grounded in the action, process, object and schema themes.

4.5.5 Interviews

After completion of the interventions as indicated above, the researcher endeavoured to determine if the learners developed the ability to link APOS theory with mathematics and programming.

4.5.5.1 Interviews IA and IB: Algebra exercise on simplification; Science assessment question and Voltage-Ampere-Resistance pyramid (Appendices R-1, R-2, R-3 and R-4)

Interviews IA and IB were conducted to explore and understand how the learners' thoughts were ordered when assessments in Mathematics and Science were revisited. The interviews were conducted in an informal environment and open-ended questions were asked to explore the thinking of learners on any mathematical problem of their choice. The researcher provided several mathematical problems that learners encountered at the time and gave the learners the choice to select any of them.

4.5.6 Observations

During observations, the researcher took notes on how learners reacted during interventions and interviews. These observations were also supported by taking videos where possible. The videos tended to distract the learners' attention when they were busy with certain programming tasks. Observations were conducted throughout the research process, supported by notes. Reflections led to dual outcomes contained in (i) a practical real-world contribution through abduction found in an ongoing sub0cycle of design-reflection-design, and ii) a theoretical contribution where a new theory was developed through multiple EDR cycles to reach maturity in the form of a conceptual framework.

4.5.7 Data collection strategies

For this research, the researcher explored the role of computational thinking among high school learners and how computational thinking could be promoted among high school learners. Data were collected from the interventions, the informal interviews, and the observations. Each intervention was designed to collect data from the learners. The intervention data were used iteratively, as some intervention outcomes served as the input for the next intervention. Depending on the intervention, the data collection adopted different techniques, such as written assessments, projects, programming and interviews. The interviews and outcomes of the assessments and assignments were transcribed (Appendices R1 to R4) and provided insight into learner perceptions on concepts such as abstraction. Four (4) Interviews were conducted informally based on a questioning plan in a conversational style. The interaction of learners with one another was observed during their PC and programming language usage. Informal notes (Appendix R-3, R-4) were made by the researcher as the interactions proceeded. These interactions were also captured on video.

To view the videos, copy the following link to a browser or click on the following Microsoft Teams link:

https://teams.microsoft.com/_#/school/files/General?threadId=19%3A8eff649f6647440ea80e6841eb4dcce8%40thead.tacv2&ctx=channel&context=NavorsingsVideos%2520Curro%2520WCED&rootfolder=%252Fsites%252FResearch742%252FShared%2520Documents%252FGeneral%252FNavorsingsVideos%2520Curro%2520WCED

The coding produced by the learners acted as documentation that showed their competency skills, highlighting strengths and weaknesses of the programming activities, supported by an analysis of assessments on these programming activities (Appendix G to P). Assessments on the programming language concepts also contributed to the data collection (Appendix M-1, M-2). The researcher's interaction with the same learners spanned across two years, during which lessons were prepared by the researcher and tasks were given to the learners. These were based on genetic decomposition as introduced by Dubinsky (Arnon et al., 2014) as part of his APOS theory to obtain an understanding through observation, assessments and interviews. Data were also collected through lectures, and assessments and tasks given to learners.

4.5.8 Sampling

According to Sharma (2017:749), sampling is a "means by which researchers may select a subset from the total population to act as a data source for experimentation or observation to achieve the objectives of the research". A non-probability purposive sampling method was used to select the sample consisting of research participants for this qualitative research (Maree, 2012). This sampling method could further be categorised into several of which the homogenous sampling scheme was used for a specific group of learners having the same

characteristics when considering their learning profiles. Although Sharma (2017) regards this method as highly likely to be prone to researcher bias, the selection of the two classes, consisting of one Afrikaans group and one English group of learners, was left to the teacher dealing with timetabling of the school. Thus, the researcher was not responsible for the selection and the researcher bias, as pointed out by Sharma (2017), was removed. The only judgement of the researcher in using this sampling technique was made in terms of the ages of the learners. Learners had to fall within the cognitive level of formal operations phase, which coincided with the ages of learners in grades 8 and 9 of the private school (Piaget, 1964; White, 2003; White & Sivitanides, 2002; Young, 2011; Cherry, 2014; Ghazi et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017).

The sampling method aligned well with the theoretical conceptual framework of this research. The teacher chose classes based on the availability of time slots to conform to the practical day-to-day operation of the school without disrupting classes. To mitigate researcher bias, as mentioned by Sharma (2017), the teacher did not make any groupings on the basis of performance, but selected two classes from the private school based on convenience sampling. Grade 8 learners were also chosen, as no real programming education took place at that point in time, which minimised any programming habits and concerns of beliefs (Cegielski & Hall, 2006; Moscucci, 2007; Moscucci & Bibbo, 2015) that might already have existed about mathematics in each learner. According to Piaget (1964), a factor that may influence cognitive development is maturation when referring to the central nervous system of the learner. This is influenced by cultural differences, experiences with objects in the physical world, social transmission and equilibration when a learner reconciles his/her experiences through assimilation or accommodation (Cherry, 2014; Ghazi et al., 2014; Arnon et al., 2014; Barrouillet, 2015; Bormanaki & Khoshhal, 2017). The unit of analysis was identified as the programming language used in the research. The unit of observation was the learners participating in the research.

4.5.9 Data analysis

Miles, Huberman and Saldaña (2014) regard qualitative data analysis as collecting data and then embarking on a process of (i) data condensation, (ii) data display and (iii) drawing conclusions or performing verifications. This can be positioned within the EDR method for each intervention that was initiated. Data condensation is also an iterative process in terms of summarising data, coding, developing themes, generating categories and writing analytic memos (Miles, Huberman & Saldaña, 2014:12). The data were coded and the findings were summarised, as shown in [Table 5.13](#). Once this was done, categories were identified. The categories were then grouped into the themes of the study ([Table 5.13](#) & [Table 5.15](#)).

4.6 Summary

Chapter 4 addressed the research and design of this study. The following concepts were discussed: (i) the demonstration case; (ii) action research; (iii) DR strategy; (iv) intervention development; (v) interviews and observations; (vi) sampling; and (vii) data analysis. The research approach was focused on conducting research through abduction. Research strategies were considered to explore the role of computational thinking among high school learners and how computational thinking could be promoted through a programming language using APOS theory as lens, at a cognitive level of formal operations.

AR as a research strategy did not satisfy the goals of this research, which included producing an artefact; and social or organisational change was not part of the outcome. Didactics, the learners' involvement, mathematics, hardware IT platforms and programming attributed to theory building. All these components played a role in the outcome of this research and strengthened the scientific value of AR was limiting compared to EDR in terms of processes and outcomes. EDR was chosen as research strategy. This research focused on the science of the artificial, also known as Design Science (DS). DS focuses on phenomena that serve human purposes. Thirty-four (24) interventions were executed for this study. In many ways, the interventions fed into each other. A private school was chosen as case, with the programming language as unit of analysis and the learners of the school the unit of observation.

5.1 Introduction

Figure 5.1 represents the layout of Chapter 5. The research was done on the effect of a programming language as a vehicle, with APOS theory as lens to influence the learner's computational thinking towards subjects which require the cognitive level of formal operations, such as Mathematics or subjects that demand complex learning. Figure 5.2 illustrates these keywords used in this thesis, capturing the essence of the study. Greenfoot as programming language was not used to predict a change in mathematical assessment outcomes of learners. The goal of Greenfoot is to promote computational thinking through APOS theory among learners or change their computational thinking that may influence their system of beliefs about mathematics. By addressing computational thinking, the thought processes (Selby & Woollard, 2014; Denning, 2017) of learners are influenced. It is widely accepted that changing learners' behaviour, teachers' practices and their (learners and teachers) approach to mathematics, need to change in both parties' beliefs about mathematics. This is a long-term goal as opposed to a 'quick fix' (Moscucci, 2007; Moscucci & Bibbo, 2015; Jankvist & Niss, 2018).

5.2 Belief system about Mathematics

This research explored and aimed to understand the effect of Greenfoot as programming language on computational thinking using APOS theory as lens. The research made use of EDR to employ interventions among grade 8 and 9 learners sequentially. These interventions were interactions with the Greenfoot programming language and positioned the learner as a candidate computational thinker, in line with the APOS theory in mathematics. PLs are a fresh and relatively unknown domain for most learners and teachers as pointed out by the learners upon introducing the goals of the interventions at the time of this research. Mathematical thinking that accompanied learners for twelve consecutive years created a belief system about mathematics for each learner and each teacher. Every grade allows for a new teacher with different beliefs about mathematics. According to Moscucci and Bibbo (2015), belief systems require more research, for beliefs, attitudes and emotions are interconnected. Moscucci (2007:1) describes the process as "learning of beliefs", which entails learners' personal mathematical approach and that of the teachers' beliefs about mathematics as influential for any learner, when forming a system of beliefs about mathematics. LeDoux (1998) and Damasio (1999) describe this with their cognitive-emotional structure. Moscucci (2007) put forward the meta-belief system activity (MBSA), which is a framework used to rebuild negative relationships with mathematics, but involves attitudes, emotions and cognition towards mathematics. Moscucci and Bibbo (2015) emphasise that mirror neurons (feeling the mind state of another human) play a role in imitation and action understanding as highlighted by Rizzolatti and Craighero (2004). The authors further see communication as visible and tangible components, but also as

unconscious and invisible components. It is this invisible and unconscious or hidden communication that plays a role in beliefs about mathematics. Nelson (2012) sees an example of hidden communication as empathy. Research of Moscucci and Bibbo (2015) shows that learners have pre-set ideas of what teachers think of their ability developed through these invisible components or mirror neurons.

Bachelard (1938), Brousseau (1983, 2002), Moscucci (2007), Jankvist and Niss (2018) argue that existing beliefs about mathematics around mathematical education, beliefs about the self, the social content form part of a difficult domain to penetrate and to change in an instant, from a negative to a positive system of belief about mathematics. Belief systems about mathematics within the sample group of learners were not specifically assessed using the meta-belief system activity (MBSA) framework of Moscucci (2007). This MBSA is built on meta-cognition as introduced by Flavell (1976) and transformed into a tangible framework by Moscucci (2007). Metacognition is broadly defined as “thinking about thinking” (Moritz & Lysaker, 2018). Livingstone (2003:2) quotes Flavell (1979) that metacognition is “higher order thinking which involves active control over the cognitive processes engaged in learning”. This research studied the promotion of computational thinking from an unknown perspective (programming language belief system) such as a programming language, which may act as a meta-cognitive approach, seeing that only one learner has encountered a programming language before and the research is done by controlling learners’ cognitive processes through APOS theory at a higher level. Jackson (2004) uses the term metalearning to deviate from metacognition in its basic definition, by adding a high-level of thinking about learning and how an individual acquires new knowledge through more effective learning.

The research also highlights the challenge of the didactical contract, by Brousseau’s theory on didactical situations, impacting the epistemological fraud, especially where learners’ and teachers’ beliefs about mathematics affect the didactical contract (Moscucci, 2007; Jankvist & Niss, 2018). For example, learner X (Appendix B-2) does not trust his belief about mathematics that the sum of the inner angles of a triangle is 180 degrees. Viewpoints of learners depicted in [Figure 5.3](#) and [Figure 5.4](#) show that they doubt their mathematical knowledge about these basic facts. Without moving blame from the learners, it is likely that the teacher was unaware of the APOS theory (from observation and conversation). The teacher could have rectified this by using Actions (Tall, 2008:9; [Figure 2.13](#)). This would have allowed the learner to move past the Action stage, to begin with a Process stage towards building Objects and Schemas for Geometry. It may be negative emotions experienced by learner X and Y that created a doubtful mind as well as the teacher’s ignorance about APOS

theory. The learner-teacher-mathematics relationship may not be a positive one towards mathematics and geometry in this case.

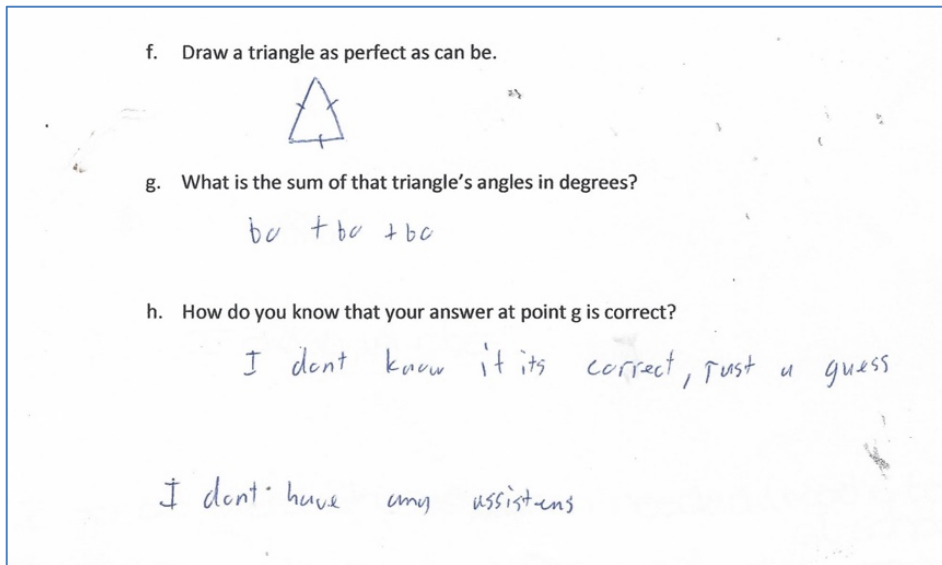


Figure 5.3: Learner X answers questionnaire on geometry knowledge (Appendix B-2)

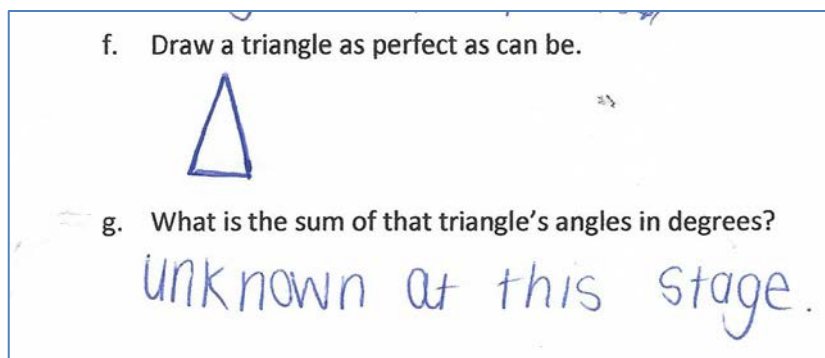


Figure 5.4: Learner Y answers questionnaire on geometry knowledge (Appendix B-3)

Questionnaires were handed out to the learners (Appendix B-3). From their responses, (Figure 5.4) it became evident that some challenges were experienced by learners. Learners were handed out the questionnaires (Appendix B-1, B-2, B-3, & B-4) and observed when they filled in these questionnaires. According to Burton (2004), most mathematics learners do not see mathematics as a creative subject, as indicated in the Figure 5.5 snapshot taken from a questionnaire (Appendix B-4).

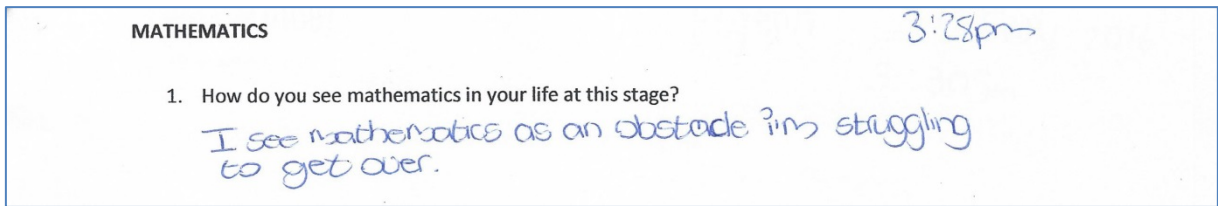


Figure 5.5: Learner Z sees Mathematics as an obstacle Appendix (B-4)

The research puts forward a new programming language belief system that influences mathematical thinking implicitly through APOS theory. The programming language belief system impacts the belief about the self in a positive manner and beliefs about the social context of how learner and teacher can function within their own world. Furthermore, in mathematics, a problem only has one right answer (Schoenfeld, 1989). When learners attempt a problem in mathematics, their answers might be incorrect which differ from their perception. According to Eisenkraft and Eisenkraft (2011), the education community may view learners' answers differently. However, a solution of an algorithm, rolled out using a programming language, can be verified immediately by learners or teachers when comparing the outcome with the result of execution. Although the coding may differ, the outcome should be the same. When referring to the same outcome, it is meant that the same goal is achieved, but the programming techniques might not be that elegant compared to other learners' programming code. As an example, learners may be asked to print the string "Hello" 20 times on the screen. Some learners may use a "print statement" 20 times and others may employ a "for"-loop structure consisting of two lines of code. In both cases "Hello" will appear 20 times, which shows the same outcome, but the code tells a different solution.

The learner can verify his or her code as correct or incorrect based on the outcome. The learner cannot always verify his or her answer in mathematics as correct, but in a programming language these learners can use a technique called debugging to trace their syntax to be logically correct. Learners enjoy an embodied experience when developing a computational model for a problem in the Greenfoot programming language (section 5.4.3.32, intervention 14D, Appendix P-4). Within (embodied) coding the learner can experience different forms of abstraction to implement a more elegant coded solution for the problem. Although the outcome of the solution looks similar, the algorithm developed by the learners is different and allows for free thinking. This varies from learner to learner. The research method, as investigative tool motivated in section 3.2.3, is EDR by intervention.

The research further focused on the exploration and understanding of computational thinking, which affects the mathematics dilemma that South Africa is facing (section 3.2). This dilemma is because of learners shying away from taking Mathematics and Science

subjects, and those brave enough to take Mathematics and Science subjects, struggle with a very low throughput as highlighted in the literature review in Chapter 2. The next section focuses on the EDR phases used in this research.

5.3 EDR phases

Having considered the EDR phases of Reeves (2006), Mckenney and Reeves (2012), Bannan, (2013), Plomp (2013), van Wyk and de Villiers (2018), Miah, Solomonides and Gammack (2019), as described in section 3.2.3.3, the following three generic EDR phases depicted in section 3.3.4, are integrated with the van Wyk and de Villiers (2018) model. The van Wyk and de Villiers (2018) model describes processes and outcomes for each process. It is also an inferred model that stems from DR, DSR and DBR (section 4.5.2, Figure 4.8). This research has a diverse focus on education with a mathematics specialisation programming language and coding and information systems with an emphasis on systems development. Two important phases within the van Wyk and de Villiers (2018) model are evaluation and reflection. Evaluation is investigated through the FEDS framework (Venable, Pries-Heje & Baskerville, 2016) and the reflection phase through the framework of Gregor, Müller and Seidel (2013).

5.4 Data collection and analysis

5.4.1 Introduction

The phases are integrated with the van Wyk and de Villiers (2018) EDR model and are discussed in the following sections.

5.4.2 Phase 1: Preliminary research phase

This phase requires preliminary research that entails a needs and context analysis, a literature review, theory development, and the identification of a target group (Figure 3.10, section 3.3.4). The model of van Wyk and de Villiers (2018) describes the process as the analysis of a complex problem (Figure 5.6). The process of the problem analysis yields outcomes, such as research goals which contribute to the research proposal.

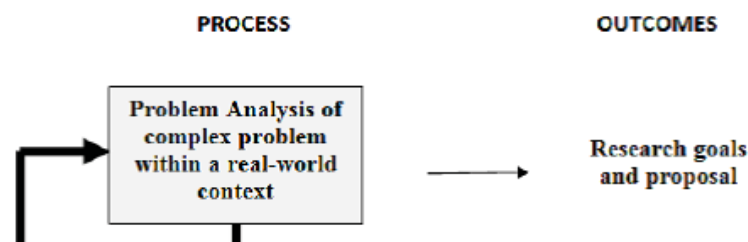


Figure 5.6: Cross Section of EDR model (Adopted from Van Wyk & De Villiers, 2018:305)

The problem is identified by doing a needs and context analysis which is augmented by a literature review and which is an interpretation of the literature review in the traditional research track. The FEDS framework (Venable, Pries-Heje & Baskerville, 2016) and the framework of Gregor, Müller and Seidel (2013) are used to accomplish the research goals and proposal. This is followed by theory development and the identification of the target group (Chapter 3, [Figure 3.10](#)) discussed in the next sections.

5.4.2.1 Needs and context analysis

(a) Needs analysis

Learner performance in mathematics is deteriorating in South Africa. The warning lights are ignored up to Grade 9, and learners are left to partake in extra mathematics classes. These learners pay exorbitant tuition fees to mathematical powerhouses (MPHs) to strengthen their concept images of mathematical concepts (Arnon et al., 2014) where education should be a basic right. Having identified the need to address this problem in education, the next section highlights the need within the context of mathematics and a programming language.

(b) Context analysis

The concept images allow learners to answer mathematical papers during examinations and adhere to DBE minimum standards (Chapter 2). Although there is a collective calling from the DBE, national and international companies are making money off the mathematics dilemma in South Africa. One such example is <https://za.ixl.com>, to mention one website that adheres to CAPS in SA. As a side note, researcher rolled out the <https://za.ixl.com> site among three schools and compared learners' school assessments against previous assessments after using the site. The method used on this site is based on explaining a concept, but also monotonous repetition of those mathematical concepts. According to Trends in International Mathematics and Science Study (TIMSS), in 2011, South African learners performed the worst out of 21 middle-income countries (CDE, 2014). The trends as given by CDE (2014) are rather old, but since 2014, the DBE changed from annual national assessments (ANAs) to systemic assessments (Schäfer, 2018). Unfortunately, the systemic assessments have not been substantiated as the results may hide weak performance. The impact of systemic tests can be investigated by the reader as a trivial exercise of how systemic tests may influence learners and schools. As a result, the research chose to use the official 2014 figures/ results. The DBE attempted to rectify these figures in a learner's final year of study at school. This raises questions by the researcher such as "what was done in the former schooling years of these learners towards their final year at school?" and "what is being done at this stage"? How the DBE wants to raise the throughput of learners in mathematics is a wicked problem because of their beliefs about mathematics (Moscucci,

2007) at both the teacher's and the learner's perspectives, which influences mathematical problem solving and indirectly the computational thinking skills of learners in SA.

5.4.2.2 The literature review

As the literature review was done in Chapter 2, only a brief overview is provided here, and is discussed under the headings: (a) thought processes; (b) computational thinking; and (c) APOS theory. The review of the literature provides the problem identification and the needs and context analysis.

(a) Thought processes

Characteristics such as “concept images” in mathematics, “beliefs about mathematics”, “Mathematical problem solving”, “cognitive levels”, “computational thinking”, “reflective abstraction”, “algorithms”, “thought processes”, “abstraction” and “automation” form part of the challenge learners are facing in SA.

This research focused on the cognitive levels of learners to position the interventions at an age and grade where accommodation and assimilation are successful to a degree. The cognitive levels are discussed in section 2.2.2.1(a)(i). According to Piaget (1964), Young (2012), Cherry (2014), Ghazi et al. (2014), Barrouillet (2015) and Bormanaki and Khoshhal (2017) the cognitive level of formal operations is the level where thought processes accommodate complex learning. Also see Figure 1.2, section 1.8 for an updated breakdown. Factors necessary to kick-start computational thinking at a cognitive level of formal operations are investigated through SRQs. The study shows that mathematical problem solving and computational thinking are linked through the concept of reflective abstraction. Reflective abstraction is used in the context of computational thinking (Cetin & Dubinsky, 2017). Denning (2017) interprets Aho's (2012) definition of computational thinking as the thought processes necessary to formulate problems. Selby and Woollard (2014) also link thought processes to computational thinking. The thought processes should bring about solutions to problems. These solutions can be represented as computational steps and algorithms as depicted in section 2.2.2.1(b)(iii), Figure 2.10. Selby and Woollard (2014) show that three aspects are always found in the definition of computational thinking, namely thought processes, the concept of abstraction and the concept of decomposition. The terms ‘problem solving’ and ‘logical thinking’ are too broad, which focus more on skills development.

(b) Computational thinking

Abstraction and automation are the “mental and metal tools” of computational thinking (Wing, 2006, 2008). Denning (2017) posits that following any sequence of steps or algorithm does not necessarily make you a computational thinker. Aho (2012) states that computational

thinking is about finding appropriate models of computation to derive a solution for a formulated problem. Researchers such as Hayakawa (1949), Truran (1992), Wilensky (1991), Dubinsky (1991), Hazzan (1999, 2003), Devlin (2003), (Kramer 2007), Perrenet (2010) and Meyer (2010) state subtle differences when arguing the concept of abstraction. Wilensky (1991:4) states “concreteness, then, is that property which measures the degree of our relatedness to the object, (the richness of our presentations, interactions, connections with the object), how close we are to it, if you will the quality of our relationship with the object”.

(c) APOS theory

The APOS theory originated from Dubinsky's (1991) interpretation of Piaget's (1973) concept of reflective abstraction. The research on APOS theory is based on mathematical problem solving and how learners should approach mathematics in general (Arnon et al., 2014). Piaget sees the properties of objects not in the objects itself but embedded in the actions that learners take when using these objects (Arnon et al., 2014). Each mental construction (**A**ction, **P**rocess, **O**bject, **S**chema) depicted in [Figure 2.17](#) or conception uses mental mechanisms (interiorisation, coordination, reversal, encapsulation and thematisation) to move through the APOS (mental structures) cycle. Reflective abstraction is a description of what goes on in the minds of individuals when engaged in creating knowledge. It is hypothetical as nobody can see what goes on inside another's mind (Dubinsky, 1991, 2000).

There are many possible ways to solve a mathematical problem, which may confuse the learners' actions. This often leads to lower levels of abstraction that complicates their understanding (Hazzan, 1999; Kramer, 2007). Researchers describe this state a learner is in as a state of “abstraction anxiety” (Sfard, 1991; Wilensky, 1991; Meyer, 2010), which forms an important component of mathematical anxiety. Papert (1980) uses the term ‘mathophobia’ and Tall (2004) refers to this phenomenon as ‘dyscalculia’. Meyer (2010) states that educators should not only classify subjects as being abstract, but also deal with this anxiety associated with abstraction. The next section describes the artefact as outcome from the EDR approach.

(d) The EDR question

The EDR question which describes the artefact as one of the outcomes is motivated in section [3.3.3.1](#) to determine the research goals after a problem analysis of the problem statement (section [1.5](#)).

(i) Research goals and proposal

The research problem for this thesis reads as follows: computational thinking lacks among learners because it is not clear how computational thinking is promoted at the cognitive level

of formal operations among high school learners. The research questions addressing the research problem are stated in the following two tables, for ease of reference. The objectives are mapped in [Table 5.1](#) and [Table 5.2](#) to the SRQs as a first step to create the goals for this research.

Table 5.1: Mapping of research methods and goals to Research Question 1

RQ 1	What are the characteristics of an enhanced learner’s teaching and learning strategy that can empower learners to master computational thinking skills through APOS theory, infused by a programming language at high school level?	
SQRs	Research Method	Objective
1.1: What factors are needed for the development of computational thinking at a cognitive level of formal operations among high school learners?	Literature Analysis	The objective of the question is to determine the factors which promote computational thinking among high school learners at a cognitive level of formal operations (CLFO).
1.2: What type of programming language may be used to promote computational thinking skills at a cognitive level of formal operations?	Critical analysis of the features of programming language	The objective of the question is to determine the characteristics of a typical programming language that may promote the cognitive level of formal operations (CLFO).
1.3: What constructs within the programming language facilitate APOS theory at a cognitive level of formal operations?	Critical analysis when comparing the constructs of the programming language and APOS	The objective of the question is to determine constructs in the programming language that promote APOS theory at a cognitive level of formal operations (CLFO).

Table 5.2: Mapping of research methods to Research Question 2

RQ 2	How can computational thinking skills at a cognitive level of formal operations be promoted among high school learners through the teaching of a programming language aligned to Action Process Object Schema (APOS)?	
SQRs	Research Method	Objective
2.1 How are the constructs of a programming language taught among high school learners at a cognitive level of formal operations?	EDR Study	To explore and understand how constructs of a programming language facilitate high school learners at a CLFO.
2.2 How do the constructs of a programming language align to APOS among high school learners at a cognitive level of formal operations?	EDR Study	To determine higher-level constructs within a programming language which promote APOS among high school learners.
2.3 How does the use of an LMS, as a platform for learning, aid the teaching of a programming language aligned to APOS to promote computational thinking skills at a cognitive level of formal operations among high school learners?	EDR Study	To combine the usage of an LMS and a programming language in order to assist high school learners with “worked examples” of advanced higher-level constructs in a programming language and cognitive load theory (CLT).

(ii) Outcomes

The research purpose is to develop the teaching and learning strategy to master computational thinking skills, through APOS theory, which is expected to function at Piaget’s cognitive level of formal operations, infused by concepts and characteristics of a programming language at high schools, in order to cope with the challenges in subjects such

as Mathematics and Science. Table 5.3 shows the EDR research goals. The APOS theory binds the programming language and computational thinking.

Table 5.3: The research EDR goals and objectives

Goals	Objective
To determine the factors of computational thinking at a CLFO.	The objective of the question is to determine the factors which promote computational thinking among high school learners at a cognitive level of formal operations (CLFO) using a literature analysis.
To identify the constructs in Greenfoot that enable APOS theory at a CLFO	The objective of the question is to determine constructs in the programming language that enable APOS theory at a cognitive level of formal operations (CLFO). This is done through a critical analysis when comparing the constructs of the programming language and APOS
To explore and understand the higher-level constructs in Greenfoot programming language linking into APOS theory.	The objective of the question is to explore and understand the relationship of higher-level constructs of a programming language and APOS theory during teaching and learning of high school learners at a CLFO through EDR.

(iii) Design the solution

A theoretical conceptual framework as solution (section 5.4.2.3(a), Figure 5.9) is designed (process of initial design Figure 5.7 below) to address the “teaching and learning strategy to empower learners to master computational thinking” taken from the EDR question.

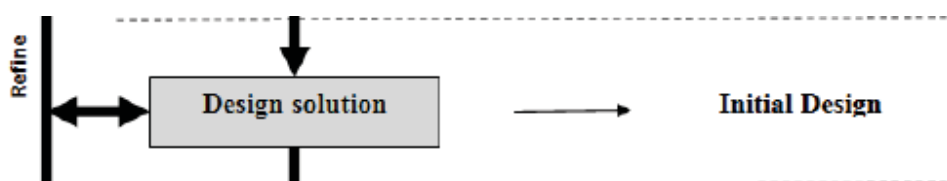


Figure 5.7: Cross section of EDR model (Adopted from Van Wyk & De Villiers, 2018:305)

The APOS theory is applicable for mathematics learning and the goal is to determine whether the learners can apply APOS theory in programming to promote computational thinking. This EDR approach is thus a validation study to validate the APOS theory within programming with similar cognitive outcomes as in mathematics learning. Over and above answering the research questions, the following learning targets were set, based on the EDR question (section 3.3.3.1) through reflection (Gregor, Müller & Seidel, 2013) namely:

- To empower mathematics and programming teachers within the DBE to acknowledge the usefulness of APOS that promotes computational thinking within their teachings to learners.
- To enable learners to understand APOS theory within their programming assignments and practices.

- To let learners apply APOS theory from a programming perspective in their mathematics learning.
- To enable teachers and learners to see the importance of an LMS as a constructionist tool.
- To regard programming as a doable subject and not as a threat anymore within the mind of the teacher and the learner.
- To increase the number of learners taking mathematics and IT.

The initial design process as depicted in [Figure 5.7](#) consists of the identification of the target group as well, as discussed in section [5.4.2.4](#). Teaching and learning of the target group is done by the researcher, who is also an instructor in Greenfoot that is supported by certification in Oracle. The first three research goals were used during the EDR approach to establish the factors of computational thinking, the constructs of Greenfoot and the constructs that promote APOS in Greenfoot. Teaching and learning the target group in Greenfoot in a stepwise process supported by an LMS. The learners must apply the goals in the Greenfoot application.

5.4.2.3 Theory development

The initial solution ([Figure 5.8](#)) shows what must be done to achieve computational thinking, but it must be supported by the theoretical conceptual framework based on the applicable theory discussed in the next section (a).

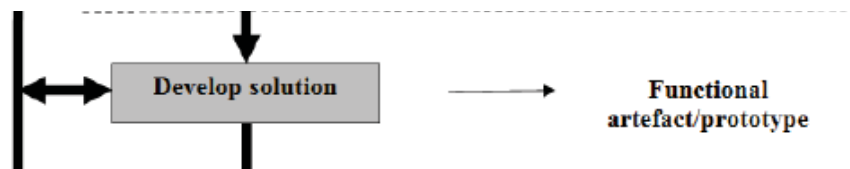


Figure 5.8: Cross section of EDR model (Adopted from Van Wyk & De Villiers, 2018:305)

(a) Theoretical conceptual framework

A theoretical conceptual framework is developed ([Figure 5.9](#)) as the initial framework for promoting computational thinking skills among learners. The theoretical conceptual framework is based on the interventions rolled out by the researcher as well as the literature study done on the problem at hand, regarding computational thinking and several genetic decompositions. The interaction of learners and computational thinking subjects such as Mathematics and Science are depicted in [Figure 5.9](#). A discussion can be found in section [2.2.2](#). For ease of reference the theoretical conceptual framework for learners is once again presented in [Figure 5.9](#).

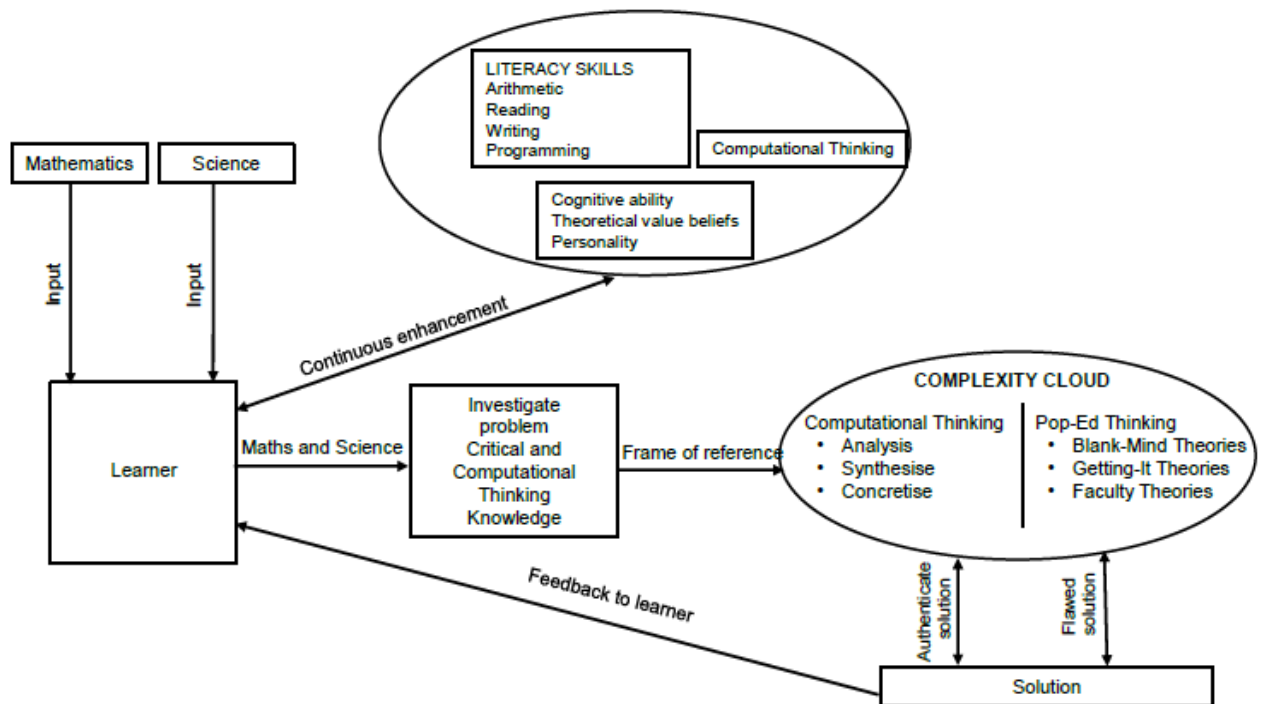


Figure 5.9: The theoretical conceptual framework for learners

5.4.2.4 Target group

The target group consisted of 18 learners of a private high school. The characteristics of these learners were determined by Piaget’s cognitive levels of development. The cognitive level of formal operations is the level which dictates the age and grade of the learner. Learners should engage at a higher order of thought processes. See section 2.2.5 for detailed information on the target group. Section 4.5 can be read in conjunction with this section to gain clarity on the demonstration case as well as the sampling process at section 4.5.8.

5.4.3 Phase 2: Prototyping/ Enactment phase

5.4.3.1 Introduction

The next enactment phase according to Bannan (2013) involves learning targets, innovation, choosing design principles, identifying and operationalising cognitive and performance processes in design, and how the design covers the theoretical model.

Van Wyk and de Villiers (2018) describe the process as “Evaluate in practice” (Figure 5.10). The outcome is the research findings, which can be found in this section under phase 2 on prototyping and enactment (Plomp, 2013).

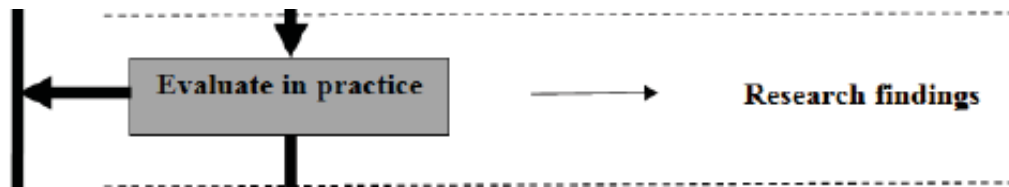


Figure 5.10: Cross section of EDR model (Adopted from Van Wyk & De Villiers, 2018:305)

Van Wyk and de Villiers (2018) correctly describe the process as an evaluation of artefacts (Figure 5.10). The findings from these interventions are mapped to the research questions (Figure 5.13, under the assessment phase 3, section 5.4.5) that support the research goals of this research.

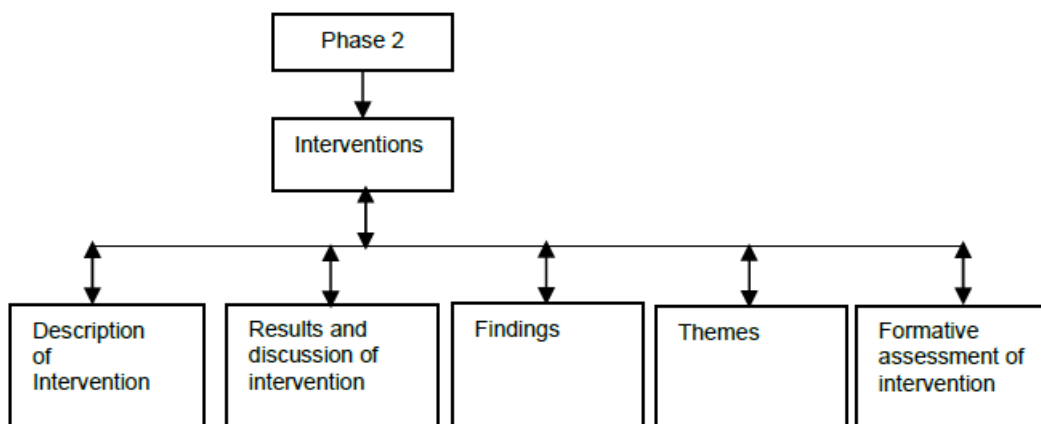


Figure 5.11: Depicts the flow of Phase 2

The needs analysis highlighted the absence of abstraction among learners and teachers not using the APOS theory within Mathematics and related computational thinking subjects. Furthermore, the basic interventions done (Intervention 1, Appendix B-1) with the learner group show (section 5.4.3.2) that abstraction as minimum requirement was (50%) absent from computational thinking. In order to conduct mathematical problem solving within mathematics is a challenge. The literature review as part of the preliminary research phase 1, points out (section 5.4.2.2) that it is not the best option to resolve the challenge within an existing belief system (section 5.2) such as a belief system about mathematics.

The formative evaluations of the interventions are repetitively criticised based on relevance, consistency, practicality and effectiveness, as depicted in Table 5.4. The abstraction forms part of computational thinking and the intervention determines whether this research should continue, or whether the abstraction skills of learners are fine. The intervention was designed around a fun element, as noticed through observation that learners enjoyed the intervention and provided feedback. The intervention is practical and applicable for the abstraction detection. The intervention is effective to provide an outcome that gives the go-ahead for this

research. The outcome is then put in perspective of the four APOS mental structures, namely Action, Process, Object and Schema. The findings (Table 5.6) show that learners have not reached a Process stage in APOS but reverted to mechanical actions in some sort or the other. It is by either physically climbing the steps or using their fingers to enact the process. These mental structures are used in mathematical thinking to perform manipulations through mental mechanisms.

Table 5.4:Criteria for high quality Interventions (Adopted from Plomp, 2013:26)

Criterion	
Relevance (also referred to as content validity)	There is a need for the intervention and its design is based on state-of-the-art (scientific) knowledge.
Consistency (also referred to as construct validity)	The intervention is 'logically' designed.
Practicality	The intervention is realistically usable in the settings for which it has been designed and developed.
Effectiveness	Using the intervention results in desired outcomes.

The presentation mode (Nieveen, 2013) of the intervention to accomplish computational thinking is built around infusing the Greenfoot programming language. Formative evaluation has led to the introduction of an LMS to enhance the quality of the learning experience, but accommodate the extra extrinsic load properly. Moodle has been used to house all the definitions on APOS and provide teacher-learner interaction beyond the notion of the curriculum. Furthermore, flipped classroom techniques as well as YouTube videos have been stored on an in-house developed and password protected Moodle website. The website can be found at (<http://wrru.co.za/moodle>), to explain abstraction and other related concepts as learners are confronted by these terms.

Teachers of the private school already adopted the tablet approach to let learners use tablets to access their curriculum per subject. The formative evaluation approach through storyboarding has been refined to bring more clarity in Moodle, the chosen LMS for the study (Bannan, 2013). The researcher, as a qualified instructional designer, created the storyboard in Moodle and adopted the text as the research progressed. The formative evaluation is determined by the FEDS framework to produce empirically based interpretations. These interpretations came from observing teachers and making notes during conversations. The evaluation is called formative, as it was done prior to rolling it out to learners.

Refinement was done by informing learners of concepts on Moodle through dynamic questioning and answering approaches, to link “met-befores” with current concepts. These storyboard techniques (<http://wrru.co.za/moodle>) make Moodle more interactive and attractive in terms of teaching and learning strategies and cognitive load theory. The content

needed to guide learners instead of having a static repository of reference material. The static presentation of most LMSs guarantees the absence of learner interaction, which nullifies the aim of an LMS and increases extraneous load (Mostyn, 2012; Sweller et al., 2019).

EDR is based on interventions to generate knowledge on computational thinking and APOS. [Table 5.5](#) provides a detailed summary of all tasks and interventions used in this research for the private school that formed the single case study, based on a legend.

Table 5.5: Detailed summary of interventions

No	Activity	Description	Appendix(A) Figure(F) Table(T)	Target Population			Consent
				Grade	English	Afrikaans	
1	Intervention 1	Abstraction (Abstract Thinking) Assessment	B-1 (A) 4.10 (F) 5.6 (T)	8	X	X	Yes
2	Intervention 2	Implement Greenfoot (Circumscribed)	C-1 (A) 4.11 (F) 5.12 (F) 5.13 (F)				
3	Intervention 2A	Introduction of Greenfoot	D-1, D-2 (A) 4.11 (F)	8	X	X	Yes
4	Intervention 2B	Introduction of a genetic decomposition (GD)	D-1 (A) 5.14 (F)	8	X	X	Yes
5	Intervention 2C	Introduction of an enhanced GD	D-3 (A) 4.11(F)	8	X	X	Yes
6	Intervention 3	Interaction with the Moodle LMS (Circumscribed)	E-1 (A)				
7	Intervention 3A	Tools to use the Moodle LMS	E-1 (A) 5.7 (T)	8	X	X	Yes
8	Intervention 3B	Juggling as the APOS example	E-2 (A) 5.15 (F) 5.16 (F) 5.8 (T)	8	X	X	Yes
9	Intervention 3C	Moodle and Generalised Terminology	E-3 (A) 5.17–28 (F)	8	X	X	Yes
10	Intervention 4	Creating a Moodle Learner Management System (LMS) (Circumscribed)	F-1, 2 (A)				
11	Intervention 4A	Moodle Learner Management System (LMS)	F-1 (A)	8	X	X	Yes
12	Intervention 4B	Creating a Cloud-based Moodle LMS	F-2 (A) 5.9 (T)	8	X		Yes

No	Activity	Description	Appendix(A) Figure(F) Table(T)	Target Population			Consent
				Grade	English	Afrikaans	
13	Intervention 5	Greenfoot Access (Circumscribed)	G1, G2 (A)				
14	Intervention 5A	Introduction to Greenfoot	G-1 (A) A-7 (A) 5.29 – 5.37 (F)	8	X	X	Yes
15	Intervention 5B	Revisit previous Activities	G-2 (A)	8	X	X	Yes
16	Intervention 6	Applying Process and Object within mathematics	H, D2 (A)	8	X	X	Yes
17	Intervention 7	Greenfoot as Process and Object	I (A) 5.33, 5.34 (F)	8	X	X	Yes
18	Intervention 8	Rollout of code in Greenfoot	J (A) 5.35 – 5.38 (F)	8	X	X	Yes
19	Intervention 9	Making decisions towards Encapsulation	K (A) 5.39 – 5.42 (F)	8	X	X	Yes
20	Intervention 10	Revisit encapsulation with Randomize	L (A) 5.43 (F)	8	X	X	Yes
21	Intervention 11	Assessment (Circumscribed)	M (A)				
22	Intervention 11A	Informing the Learners of the Assessment	M-1 (A)	8	X	X	Yes
23	Intervention 11B	The Assessment depicted	M-2 (A) 5.44 (F)	8	X	X	Yes
24	Intervention 12	The Variable in Greenfoot	N (A)	8	X	X	Yes
25	Intervention 13	Moving from Process to Object in APOS	O (A) 5.10 (T)	8	X	X	Yes
26	Intervention 14	GD creation on IF statement in Greenfoot (Circumscribed)	D-1 (A)				
27	Intervention 14A	Basic creation of scenario with World and Actor classes	P-1 (A) 5.45 (F)	9		X	Yes
28	Intervention 14B	Manipulation of Actors in a World	P-2 (A) 5.46 – 5.47 (F)	9		X	Yes
29	Intervention 14C	Interaction of Actor within the world solving problems	P-3 (A) 4.48-5.51 (F)	9		X	Yes
30	Intervention 14D	Adding graph paper as part of GD to develop algorithm	P-4 (A)	9		X	Yes
31	Intervention 14E	The IF statement as a solution to address problems	P-5 (A)	9		X	Yes

No	Activity	Description	Appendix(A) Figure(F) Table(T)	Target Population			Consent
				Grade	English	Afrikaans	
32	Intervention 15	Testing Greenfoot to be accepted among teachers	Q (A) A-8.1 (A) A-8.2/3 (A)	9		X	Yes
33	Intervention 16	Visit a group of learners outside the Greenfoot realm of instruction	U (A)	9		X	Yes
34	Intervention 17	Manufacturing Greenfoot Badges	V (A)	9		X	Yes

5.4.3.2 Intervention 1: Abstraction (Abstract Thinking) assessment (Appendix B-1)

(a) Description

For a full description see section 4.5.4.2.

(b) Results and discussion

The challenge on this intervention is infer the position of the learner after a specific number of actions were taken by that learner. These actions were stepping on and off the step. The researcher illustrated the action by physically stepping up and down on a chair. The responses are grouped and reported on per theme in Action, Process, Object and Schema. Videos such as the YouTube link <https://www.youtube.com/watch?v=webaZKOMOyU> were given to the learners, which could be pre-watched before attempting to solve the position of the person after taking 239 steps. If the learners enacted the task, as in the handout, the learner took a long time to get to the answer, exceeding 2 minutes. Some learners used their fingers to simulate climbing the steps and counted out loud as they let their fingers do the walking. Other learners applied abstract thinking and they were already able to discover a pattern within the flow of the question. They determined the answer in their minds. The solution to the problem was for the learner to use a schema of number systems embedded in his/her mind and not enacting the steps physically to determine the answer. Using their fingers is just another form of Action taken by learners.

Table 5.6 shows “Finished on Floor” answers with a total of 9. This indicates that learners either did not understand the question or in their mind understood what the question meant, but although their understanding was correct according to them, it was not in line with the concept definition of odd and even numbers. The “Finished on Step” total of 9 out of a possible 18 learners showed that 50% was able to operate at a process/object and schema level when applying the APOS theory. The Process level indicated that these learners need

not enact calculations but perform them in memory as a process. They probably also developed an odd and even number object that acted as a Schema which they used to find the solution to the problem. The certainty that these learners did not guess was the speed with which they answered the question without hesitation. The results are displayed in [Table 5.6](#) depicting the two groups of learners in grade 8.

Table 5.6: Abstract thinking among grade 8 learners

GROUPS	Finish on Floor	Finish on Step	Guess	TOTAL
A	9	9	-	18
B	9	9	-	18
TOTAL	18	18	0	36

The intervention is considered based on relevance, consistency and effectiveness.

Relevance – There is a need for this intervention seeing that it gave the researcher and learners insight in their abstraction skills. It thus contributed to the validity of content in abstraction.

Consistency – The construct is logically a success for it attracted major learner attention through enactment strategies that learners developed to find an answer. The whole class was buzzing with discussions on the problem. The example was even taken to the playground, where learners interrogated other learners with the intervention.

Effectiveness – The task is effective in that learners' thinking skills were clearly illustrated in their answers. The task also motivated further investigation into the learners' mathematical and science skills. Having done the task that highlighted the abstraction challenges of learners, the following intervention motivated the use of the Greenfoot programming language.

(c) Findings

For following discussion on the analysis of the intervention, the following findings can be stated.

Finding 1-1: Enactment in a physical format still governed the learner's thinking processes

Finding 1-2: Abstraction is a challenge for most learners and need attention

Finding 1-3: Computational thinking is a challenge for all learners

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Most learners enacted the task by using their fingers to simulate their legs, and so trying to beat the time limit. Others used their feet to enact the problem given, which is maybe worse than using their fingers. Substituting fingers for feet is already providing a form of abstraction, but it remains an enactment of body movement.

(ii) Process

A minority (less than 50%) of learners did enact in the mind as a mathematical problem associating “even” with the floor and “odd” being on the step. The mental construct of Process helped to identify that it is about odd and even numbers.

(iii) Object

The minority group of learners saw odd and even numbers as part of their being. They had the odd/even numbers as an object they could relate to.

(iv) Schema

The minority of learners used their number systems schema to isolate odd/even numbers. This triggered thought processes within the learners’ minds to deal with the problem at an abstracted level of thought.

(e) Summary

Words like minority or majority refer to less or more than 50% respectively of the learners. Having interviewed learners that got the activity correct showed that they still did not use the odd-even number system as guideline. Although the results showed a 50-50 split in the class, the correct answer was provided by the minority of learners. This indicated that computational thinking was absent to a larger extent and motivated further interventions. Questionnaires were handed out to some learners as depicted in Appendix B-1 and B-2. The learners indicated that they struggled with the current mathematical concepts, but above all had challenges with previous mathematical concepts such as the question on the sum of the triangle’s angles. The researcher assumes that the action phase that could have ensured their understanding was not done by previous teachers.

5.4.3.3 Intervention 2: Implement the Greenfoot programming language (Appendix C-1)

Having found challenges among learners to successfully answer the exercise on abstraction in Intervention 1, the findings supported the introduction of the programming language to

learners in order to assist with their thought processes in developing computational thinking (Selby & Woollard, 2014). When looking at what computational thinking encompasses, the loose interaction of learners with flipped classroom concepts brought about an enhanced theoretical conceptual framework as depicted in [Figure 5.12](#). The goal was to expose learners to a visual programming language with a gaming perspective. Introducing Greenfoot meant that learners had to adapt through accommodation or assimilation by using a programming language and develop a new belief system around programming. Programming now acted as a computational notation, but also a meta-belief system, to devise computational models, controlled by a machine (Appendix C). Prior to any algorithmic developments the learners must have had a sound knowledge of the Greenfoot IDE in general. The theoretical conceptual framework (section [2.2.3](#)) in conjunction with the literature review from an EDR perspective is updated to the depiction in [Figure 5.12](#). All the changes are shown in yellow.

The learner must be exposed to computational thinking using the Greenfoot programming language. By using the Greenfoot programming language, the algorithms that were developed based on some problem, were subject to reflective abstraction dealing with mental mechanisms in APOS theory. This was achieved by understanding and utilising Learner Working Memory optimally as depicted in [Figure 5.12](#), to reduce the cognitive load. Keeping the outcome in mind, the IDE as pointed out, remained a huge obstacle, as the IDE is the heart of the Greenfoot programming language, the controlling element from where computational thinking must be practiced. How the cognitive load can be reduced is by decreasing the extraneous (extrinsic) load (Sweller et al., 2019), as illustrated in [Figure 5.12](#).

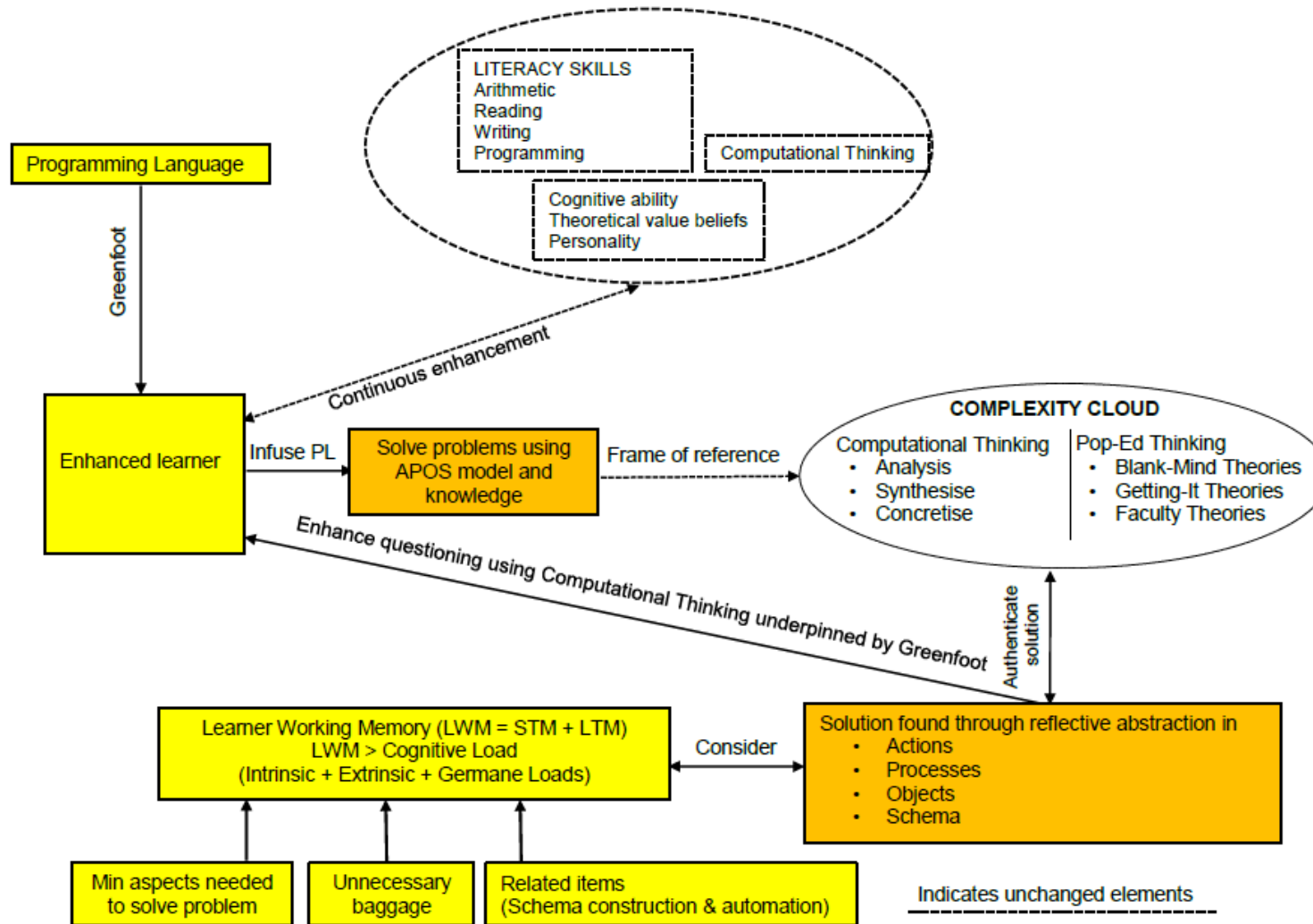


Figure 5.12: Adaptation of the theoretical conceptual framework to enhance computational thinking among learners in Grade 8

(a) Description

Intervention 2 (Appendix C) was introduced to drive computational thinking through Greenfoot as the programming language with APOS theory as lens. The goal was to guide learners towards computational thinking by using Greenfoot to act as computational notation. It was assumed that abstractions of computational models through computations can be generated by learners. The computations were supposed to run as computational models on some machine(s), controlled by an algorithm developed through computational thinking by learners. The computational model was Greenfoot in this case.

(b) Results and discussion

The Greenfoot programming language was not an obvious straight forward intervention, and sub-interventions were necessary after formative evaluations. Formative evaluations or formative assessments were done prior or during the intervention. The formative evaluations focused on learners switching-on PCs and logging into their accounts. This was followed by allowing learners to launch the Greenfoot scenario, which did not happen. This was done to cast some light on the successful outcome of the intervention and learners not spending time on peripheral activities. Peripheral activities are those activities which are of importance to let learners get to the programming language IDE, such as knowing their way around the operating system, understand the process of compilation and logging in and out. In a nutshell, it is about not having to struggle with the basics. Although learners were given videos and documentation to pre-study one week prior to the first class on how to deal with the Greenfoot IDE, the pre-study was unsuccessful. The learners just did not know what to do, which created frustration among them. Documentation included videos and steps on how to load a Greenfoot scenario were given to learners, stored on the school's shared folder on the server. Unfortunately, the learners did not pre-study or researched the documentation on starting a Greenfoot scenario. Formative evaluations were done by the researcher in allowing the learners to load a given Wombat scenario. Learners failed to find the Greenfoot programming language on their workstation, let alone opening the Greenfoot IDE.

Many researchers (Brennan, 2012; Brennan & Resnick, 2012; Papavlasopoulou, Giannkos & Jaccheri, 2019; Yu & Roque, 2019) use Scratch as a favourite programming language to grow computational thinking among learners. However, debugging and use of language constructs are important components of a programming language to assist learners with computational thinking, but they are absent from the Scratch programming language interface. Greenfoot does have that progression or path from visual coding approach to lines of coding and be a higher order language to promote computational thinking. Scratch is about boxes or shapes that represent a control structure without abiding by strict syntax. These shapes will only fit together if they belong together, which simulates strict syntax.

However, Greenfoot also makes use of a visual interface (dual modality) that can be programmed without writing any lines of code, as depicted in [Figure 5.13](#).

The formative evaluation (Gregor, Müller & Seidel, 2013) through observation was necessary, as learners could not use the IDE of Greenfoot or even reach that point. The intervention was designed from a programming perspective to implementation, but because of the challenges, such as physical enactment among learners (learners struggled to reach a Process level of the Greenfoot IDE), and they were just not ready to use Greenfoot prior to implementing sub-interventions, 2A, 2B and 2C. The “prototype of the intervention” to introduce Greenfoot (Nieveen, 2013) did not produce the desired outcome, namely the correct use of the Greenfoot integrated development environment (IDE), in that it failed as an intervention theory. Learners were unable to understand or figure out what the purpose of an IDE was. This is also the reason why formative evaluations are important to identify shortfalls prior to the rollout of any intervention. One learner commented that she is in the arts and “this programming is not for her”. Even during normal teaching and learning activity, formative evaluations are sometimes overlooked to establish the “met-before” for students to participate in a new lecture or class. The new lesson must align with previous knowledge gained. It was found that most learners (90%) were not ready to accept Greenfoot as to how, where and why Greenfoot should be used. This should hold true for any programming language used by learners for the first time (researcher’s intervention). The programming language was foreign to most students as “met-befores” were lacking and foreign to them. The single intervention of introducing Greenfoot then spiralled into three sub-interventions namely 2A, 2B and 2C as discussed further on in the section.

Intervention 2A: Introduction of a genetic decomposition (GD) (Appendix D-1)

Intervention 2B: Introduction of an enhanced GD (Appendix D-2)

Intervention 2C: Help Documentation in Greenfoot (Appendix D-3)

When studying any programming language, the habit in never stating anything about the IDE, creates its own challenge. This may also undermine further research. Journal articles seldom speak of this part where learners are not the ideal subjects and assume the subject’s knowledge about the IDE as a given. Usually this is where learner confusion prevents learners from interacting with the IDE and can be instrumental to failure in the rollout as the student commented earlier.

The introduction of the programming language (Greenfoot) called for formative evaluations to be considered such as:

- **Formative evaluation 1:** To establish a Greenfoot environment among learners the researcher must consider aspects of learner “met-befores”. When using the term environment, it refers to the IDE and installation of the product. The time spent on the IDE is a basic Action to Process activity where learners should explore the IDE and memorise how to start the Greenfoot programming language, how to compile and edit code. The installation of Greenfoot is also of significance to those learners who want to install Greenfoot on their home PCs. The researcher should consider this as pre-knowledge.
- **Formative evaluation 2:** To observe the progress of Piaget’s organisation and adaptation processes (Woolfolk, Winne & Perry, 2003; Bormanaki & Koschhal, 2017). The researcher observed the status of all the learners within the group. Through observation, the researcher became aware that some learners did not commit to any Greenfoot activity. They did not participate in working with Greenfoot, but allowed the fellow learner to take over. The researcher then assisted these learners or asked those learners that could build scenarios in Greenfoot to assist other learners. This also created a sense of urgency and collaboration among learners. Here the learners with a low self-esteem now have the chance to show those mathematics performers how to approach Greenfoot. This also addressed the Pop-Ed challenge raised by Papert (2005) as depicted in [Figure 5.13](#) as part of the theoretical conceptual framework.
- **Formative evaluation 3:** To consider the status quo of learners within Greenfoot and decide on a genetic decomposition (GD) to assist the learner to accomplish equilibration as part of the adaptation process.
- **Formative evaluation 4:** Consider and evaluate the GD and augment the GD to allow the learner to independently be able to rollout a scenario within the IDE of Greenfoot using flipped classroom techniques.

As pointed out by Denning (2017) following any sequence of steps or algorithm does not necessarily make you a computational thinker. It is about finding those appropriate models of computation to create a solution to a formulated problem directed to some machine as depicted in section [4.5.3\(a\)](#), [Figure 4.11](#).

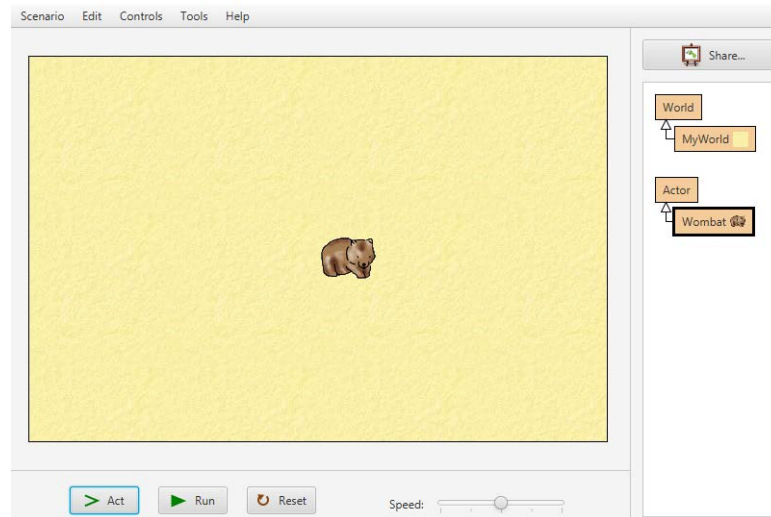


Figure 5.13: The Wombat Actor object within MyWorld World Object

The IDE embedded within Greenfoot was a challenge as learners struggled to manage ‘getting into’ Greenfoot and hence a genetic decomposition (section 2.2.2.2(b)(v), Figure 2.20) was necessary before using Greenfoot. The concept of flipped classrooms was selected as an option to utilise computer programming periods more effectively at school, which was triggered by formative evaluations among teachers or educational experts. The only difference in the technical status of teachers and learners was the local PC compared to the remote desktop (RDP). Because of RDP technology, which dictates memory sharing among workstations attached to the server, learners must make the compilations count. Making compilations count refers to learners ensuring code is correct the first time round so as to not wasting CPU cycles on the server. The RDP sessions prohibit quick compilation and learners became restless when they compile code at the same time. The flipped classroom concept was informally described to learners and they had to take it upon themselves to access videos on the Internet. The Internet was accessed by only two learners, but this was expected seeing that only two learners prepared the week prior to starting programming language classes. Although this helped to roll out the genetic decomposition of the IDE more elegantly, the flipped classroom concept needed refinement. No official research has been done to compare a flipped classroom with the outcome and purpose of a genetic decomposition. It is a combination of both GD and flipped classroom techniques. The GD acts as a prescriptive tool and the flipped classroom illustrates what is prescribed and the learner can experience learning from the specific to general.

During the formative evaluation, the ability to create scenarios was analysed through observing teachers during the Greenfoot sessions as described by the FEDS framework. The formative evaluation was naturalistic within a PC laboratory using the Human Risk and Effectiveness strategy within the FEDS framework (Venable, Pries-Heje & Baskerville, 2016). Observing especially the CAT teachers, notes were made prior to a rollout to the learners,

which compounded to experiences during class and their struggles in not knowing where to click and what option to select. The conceptual status of the teacher/student's current dilemma was still prominent, as represented in [Figure 5.12](#), when examining the complexity cloud. Although the Greenfoot programming language was introduced, the APOS theory stalled at the Actions phase and no progress was made. The learners were floating between several theoretical subjects and apart from the confusion caused by this it did not help the learner with organising his/her mind around the subjects as required by the DBE. The aim of the enhanced theoretical conceptual framework as depicted in [Figure 5.12](#) was to focus on the introduction of the Greenfoot programming language completely outside the curriculum. Having added the Greenfoot programming language to the learners' cognitive load may have caused more academic stress. A positive attitude among the learners was needed, which might have enabled learners to deal with cognitive load and enhance construction of artefacts (Papavlasopoulou, Giannkos & Jaccheri, 2019). Mostyn (2012) regards cognitive load as the mechanics of how the human brain processes data so that learning takes place. Using a GD through ACE is structured in such a way that the application of APOS theory maintains cognitive load theory.

(c) Findings

Finding 2-1: Learners do not diligently follow instructions to do homework, especially when it concerned flipped classrooms and Internet references

Finding 2-2: Only those learners with typical "met-befores" who completed the homework given to them, were keen on investigating how the Greenfoot IDE worked

Finding 2-3: The "getting-to-know" the IDE or programming language environment cannot simply be ignored and accepted as a 'given' known to learners

Finding 2-4: Learners need a GD or some proven method to clarify the IDE as dictated by formative evaluations among teachers

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Actions are taken by learners to load or run Greenfoot. Because of the existing RDP setup, these scenarios compiled and opened at a very slow speed, which sometimes made the exercise a frustrating one. Learners created a Greenfoot World and added the Wombat Actor into the World of Greenfoot. Most learners were struggling because of they did not understand the IDE, which resulted in either assistance from fellow learners or the researcher. This slowed down the goals which were set for that lesson, as the activity became collections of actions or steps that needed to be looked up by the learners.

(ii) Process

Learners were asked to watch videos on the Internet to acquire the skill of creating a scenario without performing manual steps written down. Because of the slowness of the system during compilation of the scenarios, learners were asked to view these videos. Learners performed Actions according to a physical list of steps and had no Process in place. Creating a scenario and populating the scenario with multiple Actor objects became an enjoyable exercise as they practiced more, and the Actions were slowly turned into a process. It became popular because learners experienced immediate results, but they still followed step-by-step actions to achieve the goal. Although some actions transformed into a Process, the overall usage remained action driven.

(iii) Object

The learners had an abstracted intuitive idea but no Object or thought processes yet, because they still had to refer to the physical steps.

(iv) Schema

No Schema. The conversion of the Actions taken by learners into a Process might have triggered an Object, which would have added to a Greenfoot Operational Schema. Unfortunately, learners did not practice at home, which was unacceptable, because it slowed down the research and prevented the stages of APOS theory. The researcher constantly made references to mathematics learning taking on the same progression path, so as to bring that to the attention of learners.

(e) Summary

Due to learners not participating in the documentation and videos loaded on the shared folder of the server, sub-interventions were needed to make some progress with the research. The advice and directions given to learners on what to do and which activities to follow need to be more presentable for learners to digest, as in the case of learner sessions. In order to align the presentation and requests, the following sub-interventions were suggested during formative evaluations of learners.

Intervention 2A: Introduction of a genetic decomposition (GD) (Appendix D-1)

Intervention 2B: Introduction of an enhanced GD (Appendix D-2)

Intervention 2C: Help Documentation in Greenfoot (Appendix D-3)

The EDR research question needed to be revisited in order to obtain clarity on the goal of the interventions. The research question states: "What are the characteristics of an enhanced learner's teaching and learning strategy that can empower learners to master computational

thinking skills through APOS theory, infused by a programming language at high school level?”

These characteristics prescribed a visual component to assist learners with understanding how to use Greenfoot as a programming language. Learners found it challenging to read the prescriptive GD and to translate the text into actions. Visually, through flipped classroom techniques, the learners were able to take the necessary actions to achieve the execution of a scenario. The GD was rolled out by the researcher as a prescriptive tool in presenting a lecture.

5.4.3.4 Intervention 2A: Introduction of a Genetic Decomposition (GD) (Appendices D-1, D-2)

(a) Description

The proposed GD was revised, as there were simply too many variables which the learners had to master. The goal was to create focus and make learners comfortable with the IDE of Greenfoot. The GD was supported by flipped classroom techniques to increase an understanding of the IDE in Greenfoot (<http://wrru.co.za/moodle>). The intervention was concerned with the breaking down of the Greenfoot environment such as the IDE. Learners also had to experience the Greenfoot editor and study the Help option showing the classes available and the methods per class as prescribed by the GD depicted in Appendix D-3.

(b) Analysis and discussions

The GD for “Loading a Greenfoot scenario” on the Wombat example consists of Intra-, Inter- and Trans-Wombat stages namely:

- **Intra-Wombat-Scenario Stage:** The learner is focused on the computer and Greenfoot scenario running as an application. They are confronted with looking up every action and relying on videos and the explanations of the researcher. Videos can be found on Moodle website (<http://wrru.co.za/moodle>).
- **Inter-Wombat-Scenario Stage:** The Actions of a learner to watch the video before creating a scenario is now memorised and the actions were embedded as a process. The process was encapsulated as an object called “Loading a Greenfoot scenario”. The learners separated the computer operating system and Greenfoot in their mind. This was clearly visible through observation during their interaction with Greenfoot, as they focused more on Greenfoot. Only when a scenario needed to be saved on the server, did they ask questions. The researcher explained the composition or structure of a Greenfoot scenario, so learners could understand where sound, images and source code are stored.

- **Trans-Wombat-Scenario Stage:** The schema was thematised and formed a static structure to be acted upon to “Create Greenfoot scenarios”. The schema is coherent as it guarantees success for learners when loading Greenfoot scenarios, but also allows for the learner to act on the schema to create scenarios for further development. The intervention was introduced by the researcher.

Through observation, the researcher could easily determine that learners struggled to execute the “Load a Greenfoot Scenario” as a text based prescriptive activity (Figure 2.20), although it was led by the researcher according to findings by Zeitz and Spoehr (1989) (section 2.2.2.2(c)(ii)). Through conversations with learners, they stated that they do not have internet or the Greenfoot installation at home. Videos contributed towards the learners’ understanding. The Actions to perform the activity, such as loading of a scenario, could have converted into a Process much faster if learners watched the videos. Most learners (15) did not access these videos, which lengthened the time of the research to reach the Process and subsequent phases.

(c) Findings

Finding 2A-1: Learners were confused as how to obtain access to videos on Greenfoot scenarios

Finding 2A-2: Learners still did not watch videos in general for learning purposes

Finding 2A-3: Cognitive load was still a problem in observing learners. They consulted their neighbour’s work to manage the Greenfoot programming language

(d) APOS discussion

Through observation and questioning learners remained passive learners, but their cognitive load became greater. The learners have not memorised any actions and hence no Process, Object or Schema developed.

(i) Action

The learners resorted to numerous actions which they tried to obtain from anybody in order to ‘survive’. The other mental constructions therefore do not exist.

(e) Summary

Actions were the only source these learners attempted in order to use the Greenfoot programming language. However, a GD could be used as a step-by-step guideline to assist learners to work in Greenfoot. Unlike mathematics, learners will not be able to build silos of concept images to engage in programming.

5.4.3.5 Intervention 2B: Introduction of an enhanced GD (Appendix D-1)

(a) Description of Intervention 2B

The learners were put through a GD on the Greenfoot IDE to let learners argue and think on a programming level rather than worrying about driving the application. The researcher taught the learners by having them execute the GD in a practical programming session. This was achieved by drilling editing and compilation techniques into learners when confronted with a scenario. Drilling means to repeat a basic scenario repetitively. Much more emphasis was placed on Greenfoot and less emphasis on the operating system and terminal in general. The learners opened the Greenfoot environment and remained within this environment. Files were only saved to disk on server by default.

(b) Analysis and discussion

The researcher engaged with learners as they loaded a Greenfoot application. The researcher had to assist learners throughout the class, which resulted into an operating system exercise. The importance of learners to acquire the skill to engage correctly and quickly can speed up the goal of the research by letting the students engage in the programming language through writing algorithms. Through observation, the researcher noticed that the learners struggled to create a scenario and to get it to function.

Through observation the researcher furthermore noticed that the learners' attention on programming was distracted by several unrelated activities that caused stress. These aspects, distant from programming, included:

- The workstation itself as a tool, and the slowness through an RDP session that prevented the learners from accessing Greenfoot
- The Greenfoot programming language and the IDE. The learners had to make the mind shift that the IDE is an aid towards constructing algorithms or code
- How a scenario is saved to disk. Learners had to save the scenario on the server
- The *My Documents* folder on the workstation and that of the server became confusing to the learner, for although the names were identical, the path is different. This led to the learners struggling finding the scenario to start a fresh Greenfoot application
- The running or execution of the scenario
- The loading and execution of an existing scenario
- Installation of Greenfoot from scratch on the PCs where the programming language was not yet installed
- How the scenario executed and the reason for taking such a long time. As mentioned before, the RDP sessions were dependent on shared memory on the server, influenced by all learners working on the server

The abovementioned challenges required learners to find help somewhere quickly and in a ubiquitous manner without waiting on the researcher or fellow learners to assist. The GD was augmented by adding an LMS where videos could (and still can be) be watched to facilitate a repetitive approach for learners who needed to re-visit such a GD, as depicted in Appendix D-2 (<http://www.wrru.co.za/moodle/course/view.php?id=17>).

The theoretical conceptual framework was adjusted (Figure 5.14) to accommodate the Moodle LMS, which addressed the needs of learners when they searched for videos and code snippets. Moodle also minimised the cognitive overload of the learners, as they could find the necessary resources to create Greenfoot scenarios. The addition of Moodle further addressed the complaints used as an excuse not to participate in flipped classroom techniques. Any excuses of learners were set aside by directing them to Moodle, thus addressing the unwillingness of learners when the researcher requested them to watch videos in association with the GD. This called for new rules to be implemented in using the Moodle LMS. These rules were that the researcher made it imperative for learners to visit the Moodle site and watch these videos. More important was to convince learners that the LMS was a source of information that would assist them fast in providing answers to most of their questions. A few test runs were executed through asking a question and then visiting the LMS to locate and watched the video to find a solution to the question. The videos were all placed in one location, preventing learners from visiting unknown sites in their search for Greenfoot videos. There was no need for learners to access the shared drive on their school server anymore. The interventions were assessed based on relevance, consistency and effectiveness, as depicted in Table 5.4 in section 5.4.3.

Relevance – For this intervention it was needed to include the Moodle LMS, as it positively addressed the learners' inability to implement a Greenfoot scenario, which was the starting point of this research. It contributed to the validity of content in an enhanced GD.

Consistency – The construct is logically well designed, based on the GD definition of intra-, inter- and trans-stages as a didactical construct. Consistency is obtained by adding a GD to the theoretical conceptual framework depicted in Figure 5.14.

Effectiveness – The intervention was repeated five times, but learners realised that they could perform these actions on their own with the aid of Moodle. Learners could visit and execute the steps as many times as they required for their understanding. This promoted self-confidence that the tasks were do-able. Some learners progressed further than were expected of them. This was detected through observation. The researcher could easily see whether the learner was creating a scenario or simply haphazardly clicking all options available.

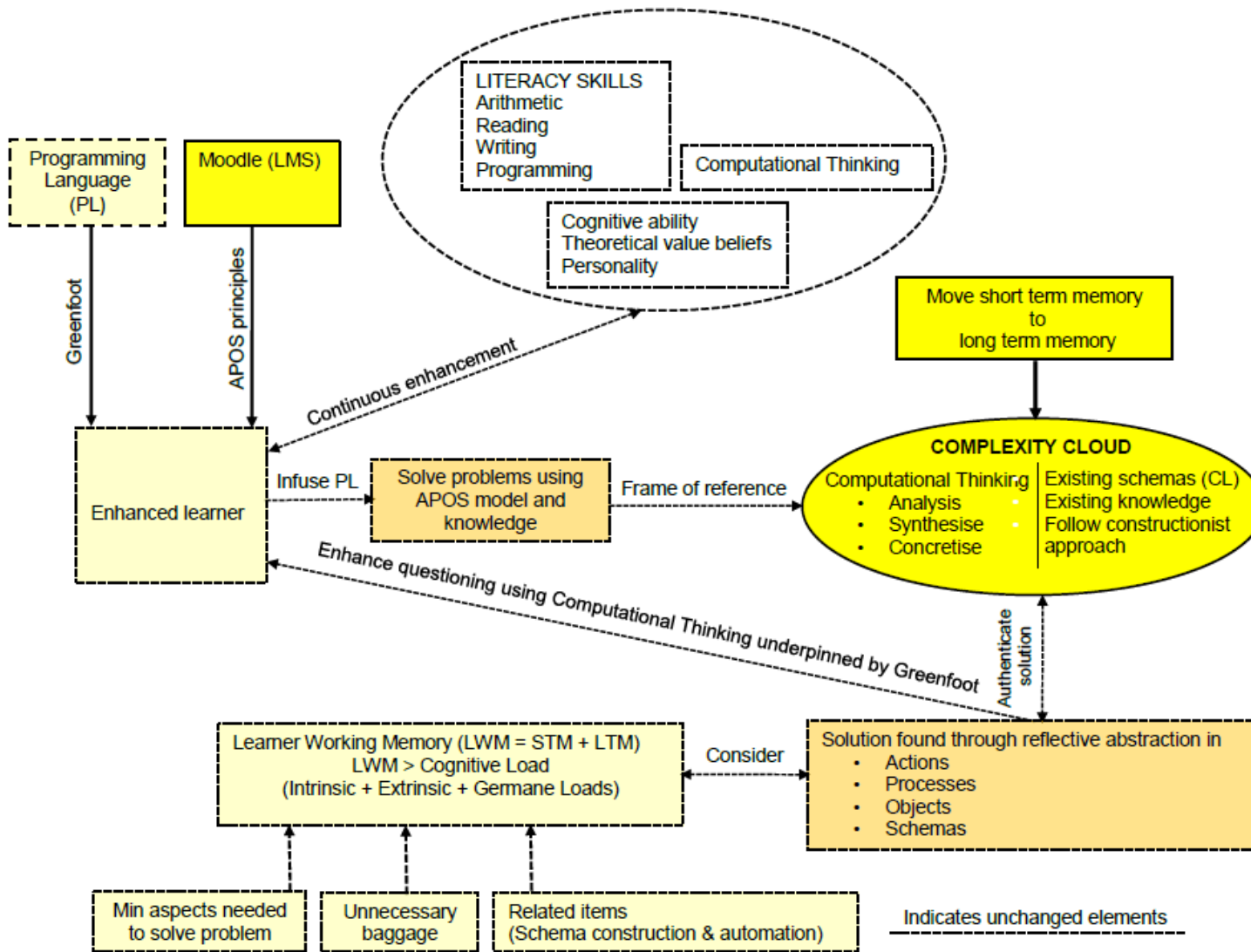


Figure 5.14: The proposed theoretical conceptual framework for enhanced learning using programming language and LMS

The criteria on relevance and consistency were met through quality interventions, based on the GD. The consistency was achieved through introducing a GD as formatively evaluated by the teachers who attended the Oracle sessions. The framework above (Figure 5.14) is a stepwise outcome for educators, which has practical implications for educators and learners as expected by design.

(c) Findings

Finding 2B-1: The Moodle was well received in that learners could access the platform in class

Finding 2B-2: The cognitive load is more relaxed as learners had a repository where information could be found

Finding 2B-3: The researcher had to impose strict measures to ensure that learners use Moodle. These measures imposed by the researcher were challenging the learners to see who could login quickly and find a topic. The researcher issued a fizzer sweet during class when a learner managed to find the answer in Moodle

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

The learners quickly remembered their login and passwords, which was their learner ID, and they devised their own password.

(ii) Processes

After logging in and out, some learners returned with lost password requests, but the researcher made them record it in a safe place. The process in waiting for a new password enforced the learners to memorise their password and login procedure. The process was incentivised by giving learners fizzer sweets when they performed well.

(iii) Object

The learners now developed Moodle as a source of information for research as an object. The term 'login; described the object, which caused learners to login without asking for help or assistance.

(iv) Schema

The viewpoint of the learners now changed in having Moodle as a partner in their educational process. This could be observed in class when they were confronted with a challenging aspect, upon which they quickly went to Moodle.

(e) Summary

The Moodle LMS was introduced to reduce cognitive load; extrinsic load to be specific. Learners managed to build a schema towards finding solutions to challenges when they engaged with the Greenfoot programming language.

5.4.3.6 Intervention 2C: Help documentation in Greenfoot (Appendix D-3)

Intervention 2C was a starting point to understand the need among the designated group(s) of learners, how the “Help” menus could assist learners in their quest to become self-autonomous. This was a typical constructionist approach but built on a truth basis, as proposed by experts in the field.

(a) Design

The learners were given exercises to explore the “Help” menus in order to better understand the Greenfoot programming language. Greenfoot consists of different classes grouped in libraries, which required from learners an understanding on how to use these structures in order to solve problems and develop algorithms.

(b) Method

Learners answered a questionnaire regarding the component structure of Greenfoot, such as the classes to be found in Greenfoot, among others. They could look it up using the terminal. This triggered the next procedure, Intervention 3A, which aimed to ensure a trusted source of information and counter the relative epistemological dilemma caused by constructivist approaches.

(c) Interpretation

The learners were observed and assessments (Appendix E-3) were given to them in order to interpret their understanding of the Greenfoot family of classes.

5.4.3.7 Intervention 3: Interaction with the Moodle LMS (Appendix E-1)

Based on the previous intervention (2C), Moodle LMS was not unconditionally accepted as a source of information, but through forced guidance by the researcher. Intervention 3 addressed the Moodle LMS as a marketing strategy and communications tool to these learners. The goal was to generate faith in Moodle as a tool through which the learner could learn at his/her own pace. The only way this could happen was to ensure that assistance could be found on Moodle LMS and that it is a trusted platform. Preparation of work always had to be towards the Moodle platform as support. It was decided to state the prerequisites for using Moodle LMS and an exercise that demanded physical actions from learners; and this exercise evolved into into a process and object. This was achieved by sub-interventions of intervention 3, namely:

Intervention 3A: Tools needed by the learners to use the Moodle LMS

Intervention 3B: Juggling as the APOS example that involves enactment among learners

Intervention 3C: Moodle and Generalised Terminology

5.4.3.8 Intervention 3A: Tools for Moodle (Appendix E-1)

(a) Description

Learners were given access to Moodle and the prerequisites were emailed to them. The prerequisites included:

- Headphones to listen to videos
- Microphone/Skype to create own sounds
- PC to create applications
- Explain to each other how code/theory works
- Always do your own work and ask for help when needed
- No real homework apart from watching videos
- No race or competition
- <http://moodle.efundo.co.za> at the time (now <http://wrru.co.za/moodle>)
- Write down list of names

When using Moodle LMS (Appendix E-1), the goal was to state a policy around Moodle usage in terms of what the learner needed and what was expected of him/her as discussed in section 2.2.2.2(c). The graphics of the Greenfoot emblem was also uploaded onto the Moodle site to foster a relationship for all learners to belong to the Greenfoot initiative.

(b) Analysis and discussion

The request to adhere to the suggested list (Appendix E-1) was received well in the sense that learners were excited about the proposal, which made the classes already more interesting. Through observation, the researcher saw that learners were excited to learn in a different way. There were instances (5) where learners left their prerequisites at home. The positive attitudes meant learners wanted to attend the classes and participate in discussions. This was noticed through observing the eagerness and enthusiasm of learners who came to class. The learners quickly discovered that the prerequisites were aimed at promoting their successful learning of Greenfoot. Measuring successfulness was based on the learners using their headphones in class to watch a video and then becoming involved in watching and applying per video guidance. The researcher allowed each learner to access these video clips within a 3-minute timeframe. The prerequisites such as headphones and microphones gave others an advantage to learn on the go. Those without their headphones and microphones had to watch a silent video. The prerequisites were loaded onto the LMS as part of the instruction list but also handed out to learners in physical format ([Table 5.7](#)).

Table 5.7: The prerequisites for using Moodle

	Prerequisites
1	Headphones to listen to videos
2	Microphone/Skype to create own sounds
3	PC to create applications
4	Explain to each other how code/theory works
5	Always do your own work and ask for help when needed
6	No real homework apart from watching videos
7	No race or competition
8	http://moodle.efundo.co.za
9	Write down list of names

Learners could use headphones in class, which was a revolutionary step when the researcher observed the positive attitudes of learners. The positivity was observed in the increase in questions these learners had on Greenfoot, which was more than usual.

(c) Findings

Finding 3A-1: Learners found this a better directive for learning, as they could bring and use their ear/headphones in class

Finding 3A-2: Not all learners brought the pre-requisites to class

Finding 3A-3: Learners developed an on-demand learning (ODL) approach, to quickly finding answers to their questions on creating a scenario and not having to wait for the researcher to come to the rescue, thereby saving a lot of time with learning concepts

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Learners took action in general by bringing headphones with them to class, but not always. Learners watched videos as per topic or concept and made progress.

(ii) Processes

Watching videos on specific concepts forced ODL, and as a result learners formed processes on rolling out concepts in Greenfoot, which placed them in a Process phase.

(iii) Object

The Moodle LMS was an object with which more actions could be taken as the concepts became more complex.

(iv) Schema

The learners held the LMS object as part of their learning schema.

(e) Summary

The list with pre-requisites was received well, although it was not always brought to class. This was probably a normal learner challenge with a normal daily school routine at any school. So, it is not something specifically targeted at this research and activities. The researcher wished that the programming language would trigger an enthusiasm that would foster more responsibility to make learners remember to bring their prerequisites along. Most important is the ODL that may have far reaching effects on other subjects as well.

5.4.3.9 Intervention 3B: Juggling enactment to enforce Moodle usage among learners (Appendix E-2)

(a) Description

Having done a formative evaluation among teachers during the Oracle sessions, the juggling exercise was popular among teachers and the researcher saw the opportunity to rollout the juggling to learners as well. The aim was to involve learners with the juggling and that these learners at the same time would find it rewarding to participate, as this is a scarce skills development.

Having added Moodle in [Figure 5.14](#) as part of the theoretical conceptual framework, the Moodle concept needed some marketing and acceptance to and by learners. Moodle needs to be embedded into a learner's mind as a source of knowledge. The goal of the intervention was to promote Moodle as a resource where learning could take place. Learners needed living proof of the contribution of Moodle to their learning. The researcher used an example by Papert (1980) to teach individuals how to juggle. Learners were asked to bring three small light weight balls to school that could fit in the palm of their hands.

(b) Analysis and discussion

The researcher showed learners how easy it is to juggle three balls and asked them to do the same. The reaction was overwhelming, but the result was disastrous across the board. Learners were then directed to the Moodle page and asked to look at the video on Juggling. The video was shown in class as well and the learners needed to "LOOK AT THE VIDEO" and "IDENTIFY" Actions. The APOS theory mental structures were pointed out to learners, performing an Action in tossing one ball at a time. This should create a Process

called Juggle, which constituted a Schema, for doing or accomplishing “Juggling” as an Object. The learners agreed that they could follow a strategy on how to juggle.

Learners accomplished the task if they kept to the rules. Appendix E-2 shows the questionnaire (Figure 5.15). Learners admitted guilt of not using the Internet or Moodle upon failing (12), when others succeeded (17) when using these tools. The results are discussed further down.

Before your evaluation tick the following textboxes

- I watched the video on basic juggling
- I practiced the methods illustrated in the video at home
- I can Juggle with ONE ball in a linear movement (tossing ball up and down)
- I can Juggle with ONE ball in circular movement (tossing ball up and down)
- I can Juggle with TWO balls, using two hands (tossing ball and handing over to other hand)
- I can Juggle with TWO balls, using one hand ONLY (tossing TWO balls with one hand)
- I can Juggle with TWO balls, using one hand ONLY (tossing TWO balls in all directions)
- I can Juggle with THREE balls (tossing THREE balls with two hands)

Figure 5.15: Questionnaire example

The juggling activity depicted in the video on Moodle and per illustration in class by the researcher shows that actions are needed before the actions become a process. This then seamlessly enables the learner to perform juggling to whatever level each learner feels able. They also realised that they had to bring the juggling devices (balls) to class in order to participate in a good lesson similar to having the right equipment such as textbook and calculator to do mathematics. The juggling activity was well received, and all the learners took part in the juggling exercise.

Table 5.8: The results of Intervention 3B

Video Watched	Video Not Watched
17	12

Seventeen out of 29 learners watched the video on Moodle on how juggling can be performed. Initially, 12 learners did not watch the video, but those who did could complete the activity and advanced to juggling with two balls. This influenced the 17 other learners to also watch the videos and as a result, which the researcher found in their answers on the questionnaires handed out, the “flipped classroom” technique brought meaning to their learning. The aim of the exercise was to highlight that the activity needed actions and practice. Questionnaires (Figure 5.15 and Figure 5.16) were handed out to learner, which

revealed some statistics on their juggling. Juggling with two balls delivered an acceptable success rate (30%). Eleven learners juggled and seven learners managed to juggle two objects physically, respectively, and four learners could juggle one object at a time within the timeframe. The overall outcome and perception of the learners were that the goal of juggling could have been reached or achieved earlier if more practice was put into the activity, and they started out in the correct manner by bringing the objects to class and watched the video beforehand. Many came back later in the month, stating that they were now capable of juggling three balls (Link in Teams³). What emerged from this exercise as stated by the learners was that they forgot to bring any juggling balls to class and hence could not perform the actions, which had a direct effect on their competence and success rate. As indicated in Figure 5.16, this learner identified the activity as a Process and wanted to exercise every day. The learners also indicated that frustration was caused by not knowing how to juggle, but through watching the video on Moodle, he managed to succeed.

If you **could not tick** all the boxes, what do you think must be done for you to be able to tick every box?
Please state clearly how you will accomplish this task.

Ek sal elke dag oefen en probeer
kyk want ek verkeerd doen gedurende
my proses om te juggle.

What did you see as a problem to accomplish this task? E.g. Self-motivation, no Internet access etc.
Selfmotivering. Ek raak verstreerd as ek dit
heertyd verkeerd doen. (nie reg kry)

If you **could tick** all the boxes, why do you think you had success?
Ek het geoften en tips gekry op video
oor hoe om te juggle.

Figure 5.16: Comments on Juggling

This is also analogous in their preparation of schoolwork in general. This questionnaire and practical exercise is a step towards creating a process instead of staying with actions. The activity sparked such interest that the learners started practicing juggling during breaks and in class. One learner just copied her partner's questionnaire answers which was an exception, but disappointing. Twelve learners stated on the questionnaire that they had no Internet access at home. The researcher had to take their word on the comment that they had no internet at home, but experience shows that learners will cast the blame somewhere

³ <https://teams.microsoft.com/l/file/A18BAF8B-B99B-42FB-9150-0B0C2D46A080?tenantId=90bb22db-a73a-4971-b7d6-7ca3ef90cf06&fileType=pdf&objectUrl=https%3A%2F%2Fcutacza.sharepoint.com%2Fsites%2FResearch742%2FShared%20Documents%2FGeneral%2FResearch%20Scans%202018%2FCaptured%20-%20Curro%20Grade%208%202014%20Greenfoot%20programming%20English%20Group%20Juggling.pdf&baseUrl=https%3A%2F%2Fcutacza.sharepoint.com%2Fsites%2FResearch742&serviceName=teams&threadId=19:8eff649f667440ea80e6841eb4dce8@thread.tacv2&groupId=a426aad5-694a-4ee5-ac2f-8fae852bfb70>

else. Seventeen (17) learners stated that they had to practice more to become good at juggling. Learners became angry when they failed at juggling, but the videos gave them hope in pursuing the action. The word perseverance was used more than often. Some used the word “afraid” to make a mistake. Some (17) started to link the activity to the APOS model in associating actions and turning that into a process when they became successful at it. Others (12) saw the time they had to invest as a problem but admitted that practicing and more exercising helped with their success. Very few (2) stated that they had no success as they probably did not watch the videos.

(c) Findings

Finding 3B-1: Learners enjoyed the exercise

Finding 3B-2: Learners were interested in this activity

Finding 3B-3: Learners acquired a better understanding of APOS mental structures

Finding 3B-4: Learners wanted to put their achievement or skill on display without showing fear among their peers

Finding 3B-5: Moodle was a ‘one-stop shop’ in getting information on acquiring the skill of ‘Juggling’ and was accepted by learners as a resource

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Learners understood the importance of an Action in tossing a ball as a starting point. Learners also see Moodle as a one-stop shop to follow actions needed to complete the challenge.

(ii) Process

Learners could transform the tossing Action into a Process by eradicating any bugs in the tossing process. They need not go back to the step-by-step approach in tossing the ball.

(iii) Object

The learners realised that tossing combinations lead to the object of tossing balls. When the researcher stated the word “tossing balls” or “juggling” the learners knew what was meant.

(iv) Schema

The name “Juggling” was given to the whole concept and the learner could associate the tossing as an Object of doing, by combining these tossing strategies. A Schema called *Juggling* was born. When they now hear the word juggling, an embedded Schema will be available to perform up to a certain individual competency of tossing balls. In changing that

Schema back to a tossing Object and increasing the competency, a higher level of skillset will be created and added to their existing “Juggling” Schema.

(e) Summary

Learners enjoyed the activity to such an extent that they took the challenge to the playground and involved their peers to partake in the challenge. The learners had one advantage over their peers in that they had access to Moodle to watch the video on explaining the “tossing” of balls. The activity led to other learners visiting the class during breaks to speak to the researcher.

5.4.3.10 Intervention 3C: Moodle and generalised terminology (Appendix E-3)

(a) Description of Intervention 3C

Having engaged learners in the previous intervention called “Juggling”, learners were now more equipped to interact with the Internet/Moodle, and they had a basic understanding of the APOS Theory. The goal of the intervention was that learners think about APOS theory and then redirect the focus to Moodle and the Internet as resources. The form in Appendix E-3 was handed out to learners to complete.

(b) Analysis and discussion

Learners answered the questionnaire (Appendix E-3). The number of answered questionnaires totalled 29. The six questions on the questionnaire are now discussed.

Question 1: Explain in your own words how you would solve any mathematics problem? What goes on in your mind when confronted with a maths problem? Even if it scares you then state that!

The learners indicated that their attention was distracted from the problem at hand. Their words indicated insecurity towards mathematics. Insecurity towards mathematics is found in words such as “try”, “stress”, “hate”, “panic”, “impossible”, “confused” and “do what they want”. Overall, words such as “doubt myself”, “panic”, “difficult”, “I think”, “frightens” indicate insecurity towards mathematics. Some (12) stated “Attempt and figure out”, which indicates that the learners had no plan or method to work with. Others (3) did show some idea by using “break problem in smaller parts”. One learner stated the term “BODMAS” which is a priority listing when simplifying an equation. However, no specific methodology was given of how it will be done.

Question 2: Explain the term “computer programming” – your understanding with an example.

The answers to this question showed a lack of understanding what programming is about. Overall, two learners showed (Figure 5.17) some understanding of the concept of programming.

2. Explain the term "computer programming" - your understanding with an example.
The coding of software that allows the user to complete a task. The program follows algorithms

Figure 5.17: General terminology

Twenty-nine (29) learners struggled to answer the question with insight based on “met-befores” and one learner linked Microsoft as the origin of all programming. Learners wrote insignificant sentences with some buzzwords they could think of (Figure 5.18).

2. Explain the term "computer programming" - your understanding with an example.
The programmes on a computer ☺

Figure 5.18: General terminology

2. Explain the term "computer programming" - your understanding with an example.
making software to run on a computer.

Figure 5.19: General terminology

2. Explain the term "computer programming" - your understanding with an example.
When you program your computer.
Like when you save programs on it.

Figure 5.20: General terminology

2. Explain the term "computer programming" - your understanding with an example.
computer running through all different files

Figure 5.21: General terminology

There was no true understanding of what programming is about (Figure 5.21; Figure 5.22), which was good for this research in terms of a programming language acting as the “meta-cognitive” concept. Some learners (21) did use “coding of software”, “storing code on a PC” and “programming”. One learner used the term “giving instructions that will turn input into desired output”, but this learner was also exposed to Python programming (Figure 5.23).

2. Explain the term "computer programming" - your understanding with an example.
Nope

Figure 5.22: General terminology

2. Explain the term "computer programming" - your understanding with an example.
giving a computer instructions that will turn an input into a desired output

Figure 5.23: General terminology

Question 3: Have you done computer programming before? (Y/N). If Y(es), give me a brief background on what exactly did you do in programming. If N(o), state why not and if you think you cannot do it and why you think it's not for you.

Only one learner stated that he/she had IT as a subject somewhere and used the word algorithm (Figure 5.24).

2. Explain the term "computer programming" - your understanding with an example.
The coding of software that allows the user to complete a task. The program follows algorithms

Figure 5.24: General terminology

Another learner made a connotation with Scratch and answered with a question that if Scratch was about programming then he/she did do this before. The learner tied programming to “cat” as if the “cat” manages the term programming. The cat was an avatar used in Scratch (Figure 5.25).

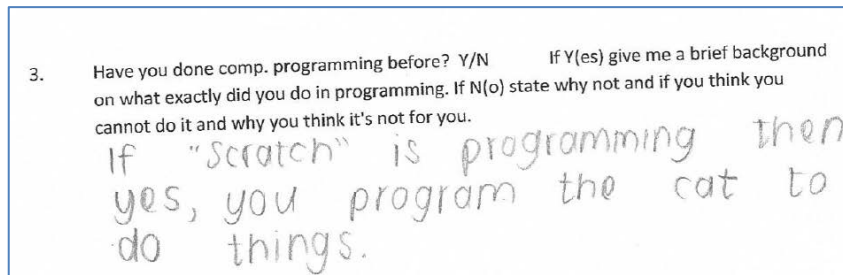


Figure 5.25: General terminology

The majority (25) stated that they have never heard of or done programming before, because of not being good with computers or that they may infect the PC with a virus, which shows their misunderstanding of programming and viruses (Figure 5.26).

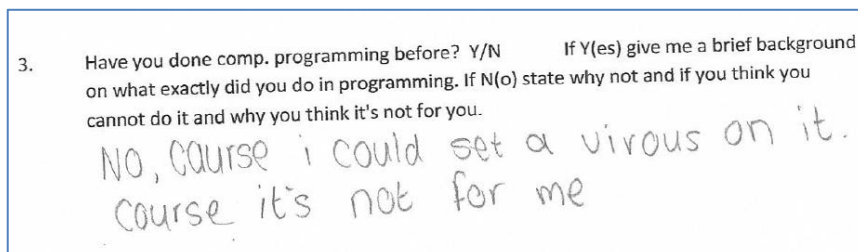


Figure 5.26: General terminology

Question 4: Have you heard of the programming language Greenfoot before? (Y/N). If Y(es), tell me what it is or how you know Greenfoot.

Learners gave an overwhelming NO as an answer (Figure 5.27).

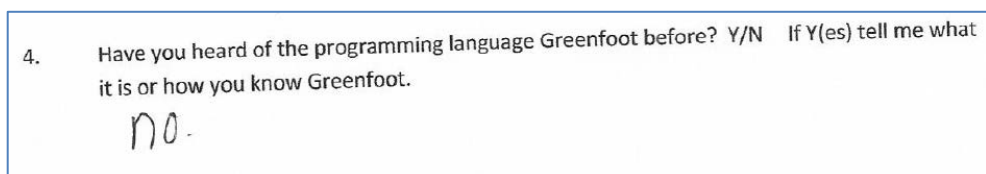


Figure 5.27: General terminology

All 29 learners have never heard of Greenfoot before. This was good from a research perspective, as a new belief system could be grounded with the intention of enhancing computational thinking.

Question 5: What do you understand by the term "ABSTRACTION"? Use any example to illustrate your understanding.

No one could figure out what abstraction means. One learner watched the videos and made some reference to a painting by Picasso, where with a few brush strokes visual meaning was

given to the painting. Another suggested it is when a rugby player obstruct another player's play/movement!

Question 6: Have you been using a technique "flipped classroom" before. Explain what you understand by "flipped classroom".

The flipped classroom technique was totally unknown to all the learners. This means that the school may not have used the technique before, or used the technique but did not attach meaning to the technique for their learners (Figure 5.28).

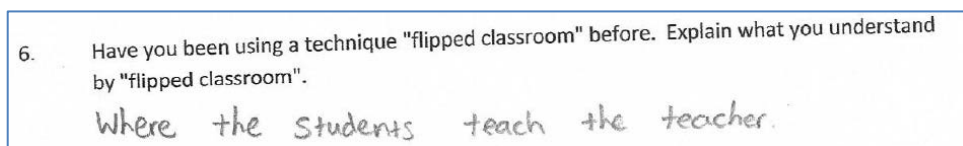


Figure 5.28: General terminology

(c) Findings

Finding 3C-1: Learners did not make a connection between APOS and mathematics

Finding 3C-2: Learners did not understand computer programming

Finding 3C-3: Learners have never encountered Greenfoot programming language

Finding 3C-4: Learners had no idea what "abstraction" means

Finding 3C-5: Learners were also unfamiliar with the term flipped-classroom

Finding 3C-6: Learners already have pre-set ideas of being negative towards programming although they did not know what it was

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Learners were reluctant to use the Internet or Moodle as a frame of reference to answer the questions. Thus, no action was taken and therefore no processes developed.

(ii) Process

No Process transformed from questionnaire exercise due to learners not taking action using Moodle or the Internet to find answers to the questions.

(iii) Object

No object transpired as no process transpired.

(iv) Schema

No Object transpired and hence no schema emerged.

(e) Summary

The answers to all the questions showed some ignorance towards having any structure in place to exhibit computational thinking. The APOS approach was simply not embedded to accommodate successful teaching and learning in subjects that demanded computational thinking. Learners might after the intervention have had a better understanding of the APOS theory, but the actual binding with the APOS Theory was absent. The acronym APOS still lacked insight and meaning for the learners. They also abandon the valuable lesson learnt in the previous interventions to use resources such as Moodle or the Internet to assist in answering the questions.

The ideal answer for Question 1 would have realised if the learner reflected on the Schemas that were put in place. The learners should have stated that the Schema needs to link with the concept definition in mathematics. If the learners had a limited Schema, they should have taken Action by finding a stepwise approach on Moodle or the Internet. These thoughts did not come to the fore and one could sense that the learners were struggling when solving a problem at this stage, even if it was just with answering these questions.

5.4.3.11 Intervention 4: Creating a Moodle Learner Management System (LMS)

Intervention 4A (Appendix F-1): Creating a Linux Server with external access

Intervention 4B (Appendix F-2): Creating a Cloud-based Moodle LMS

5.4.3.12 Intervention 4A: Creating a Linux Server with external access (Appendix F-1)

(a) Description

The introduction of Greenfoot brought its own challenges, as explained in the genetic decomposition of “Loading a Greenfoot scenario”. The goal of this intervention was to create a server with Moodle as resource accessible by all Greenfoot learners. The evaluation of this intervention is classified as a formative evaluation using the FEDS framework (Venable, Pries-Heje & Baskerville, 2016) as a Technical Risk and Efficacy evaluation strategy. This was done because schools in disadvantaged communities stated that facilities such as Moodle are expensive because it needs registration of a domain and connectivity costs. A repository was needed, which learners could access if they wanted to, but without incurring costs. Above all, the flipped classroom techniques provided invaluable support for everyone struggling with any concept as seen from Interventions 1, 2C and 3B. The configuring of a local Linux server at any school will provide several advantages, such as: no internet is needed, no extra maintenance costs are payable to a service provider, and teachers and learners have access.

(b) Analysis and discussion

Maintaining a Linux server was not the only challenge; providing external access to all users was an even greater challenge. This necessitated adapting the Moodle `config.php` file, prescribing how the Moodle site could be accessed, e.g., as <http://10.0.0.25/moodle:8090> or as <http://my.dyndns.edu/moodle:8090> or using a domain such as <http://moodle.efundo.co.za>, which now changed to <http://wrru.co.za/moodle>. The port numbers were used to create a passage from the outside into the router and allowed traffic to access the Linux box as part of the internal network in the researcher's office (house). The challenges could thus be categorised based on the degree of accessibility. The approach necessitated a study of `dyndns` setups, Linux server, setups of firewalls and router bridging setups to work with a Linux operating system.

The approach to use a local Linux box as the server and a Moodle server/repository for all interactions did work and showed that schools can setup their own service for eLearning without external interventions. Due to power failures, it was decided to create a domain `wrru.co.za` in the cloud and install a cloud-based Moodle application for better access. Part of the decision was motivated as losing the Linux server hard drive after a year and most of the work had to be recreated in the cloud. The maintenance of such a local server entailed much more than merely using Moodle when you must accomplish this task by yourself. The rollout entailed: full Linux installation, installation of Apache, MySQL and PHP. The installation of Moodle and connection of all these packages together to function as a unit, as in the case of WAMP, XAMPP, LAMP or MAMP, required much research to ensure proper functionality. To allow access from the Internet through the router into the Linux server, a `dyndns` account had to be created which, boiled down to costs for the researcher. `Dyndns` sites such as NoIP-DDNS are available for a one-time connection, which was free of charge. Being hampered by power fluctuations at inconvenient times did not ensure a perfect lecture at any location due to these uncontrollable variables.

(c) Findings

Finding 4A-1: Moodle worked from a local Linux dual Core PC. The costs of a PC can thus rather be low by using some unused PC in the school

Finding 4A-2: The Linux server needs a robust enterprise hard drive

Finding 4A-3: Learners and teachers could access the site externally

Finding 4A-4: Power failures made success intermittent

Finding 4A-5: Local content can be shared without any external connection, which relies on the LAN connection only

(d) APOS discussion

The following mental structures emerged to create an eLearning platform.

(i) Actions

The Linux server needs some expertise in terms of setup and installation.

(ii) Process

The access to the Linux server is seamless from any PC local on the LAN.

(iii) Object

Teachers and learners related the Linux server concept with eLearning.

(iv) Schema

eLearning was created as a Schema synonymous with Linux and Moodle.

(e) Summary

The Linux server can be setup as a local server within the network of any school. It can also be linked to the Internet. The school can use the setup notes of this research or use a Linux expert to configure the Linux PC. Linux runs on minimum resources and does not really need any high-performance PC. During the WCED conference (Video on TEAMS⁴) the Linux ran from the researcher's study to facilitate 40 teachers working on the Linux server at the same time. The connection was a normal ADSL 4 Mps line, using the dyndns router setup. There was no real delay experienced, and if there were no power outages on either side, the connection was solid. The learners at the private school also used the Linux server to access from their laboratory. There was an incident where power outage happened in the area of the researcher's house and prevented access. Owing to these classes happening during daytime, a power outage was minimal.

5.4.3.13 Intervention 4B: Creating a cloud-based Moodle LMS (Appendix F-2)

(a) Description

The Linux server hosted from the researcher's office 'fell over' because of the high frequency of hard drive access. It was decided that learners should still have some repository, which they could access in a ubiquitous manner, discovering more than just the textbook and classroom information. The goal of this intervention was to ensure a non-interrupted and non-eroded connection for teachers and learners. The only way to accomplish this was to register a domain and install an Open Source Moodle LMS.

4

[https://teams.microsoft.com/_#/school/files/General?threadId=19%3A8eff649f6647440ea80e6841eb4dcce8%40thread.tacv2&ctx=channel&context=2015-07-02%2520WCED%2520Kongress%2520Praatjie%2520oor%2520Moodle&rootfolder=%252Fsites%252FResearch742%252FShared%2520Documents%252FGeneral%252FNavorsingsVideos%2520Curro%2520WCED%252F2015-07-02%2520WCED%2520Kongress%2520Praatjie%2520oor%2520Moodle\)](https://teams.microsoft.com/_#/school/files/General?threadId=19%3A8eff649f6647440ea80e6841eb4dcce8%40thread.tacv2&ctx=channel&context=2015-07-02%2520WCED%2520Kongress%2520Praatjie%2520oor%2520Moodle&rootfolder=%252Fsites%252FResearch742%252FShared%2520Documents%252FGeneral%252FNavorsingsVideos%2520Curro%2520WCED%252F2015-07-02%2520WCED%2520Kongress%2520Praatjie%2520oor%2520Moodle)

The domain <http://wrru.co.za/moodle> housed the Moodle folder, depicted in Appendix F-2, where Moodle was installed and the focus was on Moodle and not on maintaining the Linux box anymore.

(b) Analysis and discussion

The learner names and their profile details were loaded in bulk using a .csv file in Excel. The process to load user profiles (learners) onto Moodle can be done within 5 minutes if a .csv file exists with all the learner names. After receiving a class list from the teacher, the file was massaged with pre-defined headings to create a profile with login and password for each learner. The researcher used a basic knowledge of Excel to massage some columns using “concat” and “substring” functions in Excel to create a file as shown in [Table 5.9](#).

Table 5.9: Moodle csv file structure

username	password	firstname	lastname	email	course1
myName1	C1111#myName1	myFirst1	MyLastName1	myLastName1@privateSchool.co.za	AGF
myName2	C1111#myName2	myFirst2	MyLastName2	myLastName2@privateSchool.co.za	AGF
myName3	C1111#myName3	myFirst3	MyLastName3	myLastName3@privateSchool.co.za	AGF

The learners now had an online email, belonged to a specific group to identify specific exercises for that group only, and had a login and password they could use to login to the LMS.

Two Moodle modules were created. One focuses on the APOS strategy looking at learners in acquiring the APOS theory and the other was on the Greenfoot language and exercises to practice examples and projects. The reason for this was the volume of Greenfoot information that might become cluttered by extra APOS information at the time. Moodle carried all the normal aspects of a learner management system, which included communication with the learners and parents in a ubiquitous manner. Learners could also change their profile photo using avatars of images that could be uploaded by the user self.

After structuring the Moodle LMS, the researcher engaged as Instructional Designer (ID). The researcher attended ID courses, as well as rolling out three modules for a private company to build capacity. At the time, the Linux server also ‘fell over’ and the essence had to be put back into the cloud. The ID concept was based on goals/objectives of a private institution that already formulated rules and standards on storyboarding their modules. The process of storyboard development happened with a team which verified the goals against the learning that took place, upon each deliverable and due date. This provided a framework that could be followed for this research. A new Moodle structure and learning activities emerged in Greenfoot with the focus on APOS theory. With Moodle structured and in place

the next intervention triggered, which is to address the Greenfoot programming language in a serious manner. Having an attractive pedagogic interface as part of Moodle LMS attracted the learners.

(c) Findings

Finding 4B-1: Moodle in the cloud had the advantage that no power outages affected delivery, unless the location of delivery had a power outage

Finding 4B-2: The cloud-based Moodle site provided ease of access ubiquitously

Finding 4B-3: Maintenance could also be taken care of in the cloud, which only needed an internet connection

Finding 4B-4: Focus on Moodle and Greenfoot development and not on a Linux server. The backups and development were a separate issue

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

The cloud-based setup needed expertise, but still required some mechanical steps.

(ii) Process

Moodle in the cloud is available 24/7. Learners knew exactly what URL to use to gain access to Moodle in the cloud. Also, login and password details had to be memorised to ensure learner access.

(iii) Object

Teachers and the researcher developed an eLearning concept linked to Moodle.

(iv) Schema

The Schema is the eLearning concept, but with a 24/7 approach.

(e) Summary

Cloud-based applications are spreading fast. Moodle is also cloud-based and accessible through the *cPanel* by teachers and researchers. Learners can access Moodle ubiquitously, through cell phone technology or browsers from notebooks, iPads or just tablet technology. Apart from being utilised as a resource, it also acts as a communication tool through gmail or any mailing systems held by learners.

5.4.3.14 Intervention 5: Greenfoot Access

Intervention 5A (Appendix G-1): Greenfoot Access

Intervention 5B (Appendix G-2): Revisit previous Activities

5.4.3.15 Intervention 5A: Greenfoot access (Appendix G-1)

(a) Description

The goal of this intervention was to provide learners with a solid foundation in how Greenfoot is accessed for programming. Moodle was installed as an LMS, so learners could download videos from the Moodle site to obtain technical assistance with creating scenarios. The Moodle site was still accessed through a dyndns setting called Rothman.for-better.biz, using port 8080. A scenario consisted of World and Actor classes within the world. The most important advantage for programmers in Greenfoot is the folder structure of resources that make up a scenario, such as the root folder with classes, sounds, images. Oracle's library on Greenfoot was also available to learners accessing it from the internet in a ubiquitous manner. Learners could now experience the whole "flipped classroom" technique and understand the benefit of having flipped classroom activities.

(b) Analysis and discussion

Owing to Java being lectured to the Grade 10 to 12 learners taking IT, the Java Standard Edition Development Kit (JDK, [Figure 5.29](#)) was already installed on the Microsoft Server, running on Hyper Terminal architecture.



Figure 5.29: (Adopted from <https://www.filehippos.org/java-development-kit/>)

The learners only needed to install Greenfoot within their user environment. Greenfoot then ran within a session on the server. Due to memory intensive operations, Greenfoot applications took quite some time to start up, which took away a large time allocation from learners during the period. Any scenario being compiled took a long time before execution could take place, which became very frustrating for both the learners and the researcher. The reason for this was that Java uses compilation of applications and all terminal output used compilation of applications on the server at the same time. This used up a lot of server resources, such as memory necessary for each Java application to execute. Other terminals then went into a waiting queue on the server regarding the output learners received from the server.

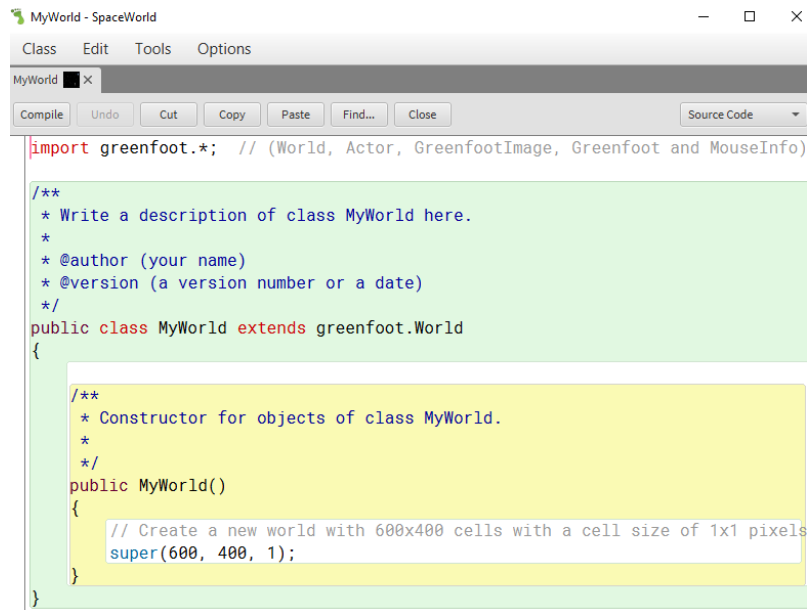
The outcome/aim of this intervention is for learners to understand what a scenario is and all the terminologies as published by Oracle. All learners had access to the Oracle library of terminologies and code. This was supported by Prof Michael Kölling's videos (Appendix A-7) (Greenfoot, 2014), which he produced and published on his site. All the videos were loaded onto Moodle as well. Many learners could do work in advance and attend the class well-prepared, as a personal choice. This was not enforced, but a friendly request. The researcher emphasised the folder structure of a scenario, consisting of doc, images, sounds and the root folder with classes.

The flow of the next activity or lesson, "what learners should investigate", was published on the Moodle server. This allowed learners to pre-prepare, which benefitted many. The term 'constructionist learning' might be a more apt description of the type of learning, as the learning entailed building practical artefacts during the research as compared to constructivist learning. The challenge with the exercise was that pushing the research to get learners into the frame of a Greenfoot mind did not suffice to the majority. Only those learners (20) that took part in every instruction through hand-outs or on the Internet or Moodle, succeeded.

EDR was the chosen strategy, which brought about a different perspective on mastering APOS theory from a computational thinking perspective, embedded within the Greenfoot programming language. The learners in the target group were subjected to constructionist learning using the Greenfoot programming language in conjunction with Moodle as a support mechanism and a resource repository to further their learning and computational thinking.

The Moodle LMS acted as a protective mechanism in hiding those learners with a self-esteem problem to investigate and participate in the transfer of knowledge with minimum teacher intervention or class exposure through questioning. The learners did not have to expose his/her ignorance about concepts though asking questions in class; they were able to research these concepts in their own time and bring knowledge to the class. It can again be compared with a flipped classroom technique. This inspired many non-performing learners to take that first step in participating in class discussions, questioning the researcher's approach with their own approach.

The "World" class uses the keyword "super", which makes use of the superclass within the Greenfoot package, as can be seen in [Figure 5.30](#). The following code ([Figure 5.30](#)) draws a rectangle of 600 by 400 cells consisting of 1 pixel each. The learner need not be concerned with the mechanics of a superclass, although this is a simple technique when they further their programming studies in Java.



```

import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class MyWorld here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class MyWorld extends greenfoot.World
{
    /**
     * Constructor for objects of class MyWorld.
     *
     *
     */
    public MyWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels
        super(600, 400, 1);
    }
}

```

Figure 5.30: Greenfoot editor with World Code

Learners could now resize their own world and create their own background, which they stored in the images folder. Those who struggled could select specific images with a specific layout as in the Actor class which contained the “Rocket” object through the “setImage” option property. All these images were categorised, e.g., Background, Transport, Animals, Food, Nature and so on, which abstracted the learner’s task. The learners enjoyed clicking on new Rocket() and created an instance that populated the world (Figure 5.31). They quickly learnt that keeping the “shift” key depressed will allow multiple instances of the Rocket object with every mouse click, as depicted in Figure 5.32.

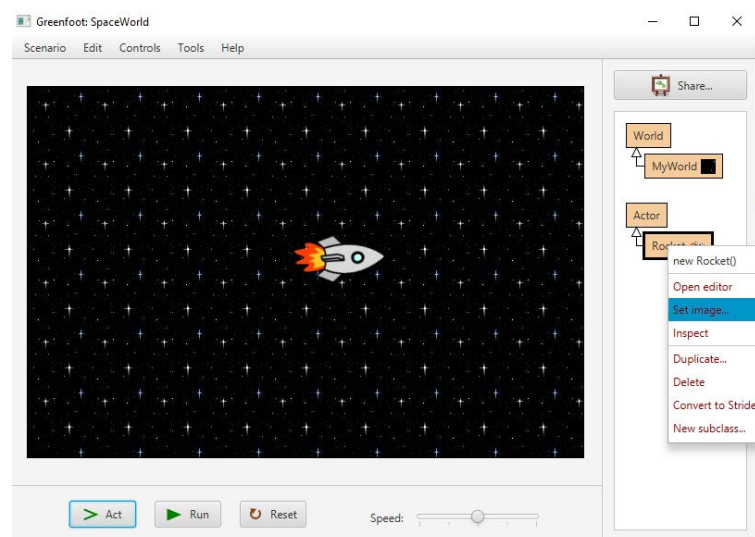


Figure 5.31: Greenfoot “SpaceWorld” scenario

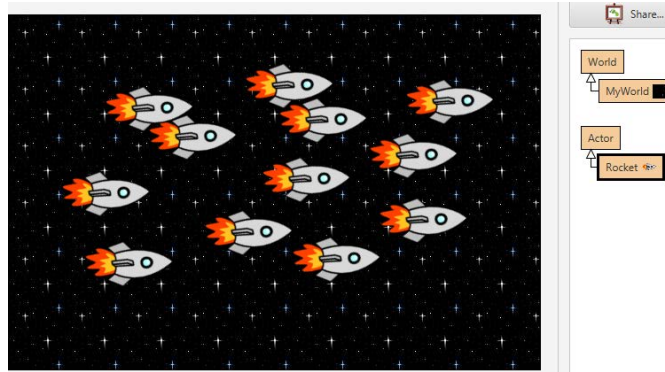


Figure 5.32: Greenfoot “SpaceWorld” World scenario populated by Rocket objects

(c) Findings

Finding 5A-1: Visual output of intervention 5A attracted attention

Finding 5A-2: Running Greenfoot in an RDP environment created memory challenges that slowed down execution of Greenfoot scenarios

Finding 5A-3: Attention span of learners became affected when scenarios took long to compile

Finding 5A-4: Stand-alone PCs may be a better option to teach Greenfoot. The usage of a VM dictates that every terminal shares memory on the server. The loading and compile time of a terminal application may take up to 5 minutes or more, depending on the utilisation of memory by the operating system

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Learners took certain steps from editor to compiler according to the GD broken down into steps.

(ii) Process

Learners managed to create a scenario and compile and run that without following steps other than that within the mind.

(iii) Object

Learners could easily relate to “scenario” as a compiled application in Greenfoot.

(iv) Schema

The Schema was created and embedded in most learners’ minds. Only the absentees tended to stay stuck on a stepwise approach in visiting Moodle to obtain the procedure in creating a scenario.

(e) Summary

Moodle was used as repository and resource to let learners adopt a constructionist approach in creating, compiling and executing the basic scenario in Greenfoot. Learners that fell ill during the lesson had to catch up extensively. Learners managed to accomplish the basic scenario and compile for errors prior to execution. Learners also compensated compilation because of the RDP environment by deciding among themselves the queuing of their compilation times. Learners did achieve Schema status by embedding the process of creating a scenario as an object in their minds. The entire approach of creating, debugging, compiling and executing a scenario was done without having to watch videos or follow a sequence of written steps.

5.4.3.16 Intervention 5B: Revisit previous activities (Appendix G-2)

The APOS mental structures in terms of the intervention are now discussed.

(a) Description

Learners had to revisit their association with APOS theory in Moodle as a resource of facts in shaping their Greenfoot approach. The goal of intervention 5B is to revisit several Action driven activities such as login procedures for learners and verifying that these Action driven approaches converted into Processes. The learners need to accomplish a number of aspects, namely: (i) login to the LMS; (ii) show an understanding of abstraction; (iii) know the APOS acronym; (iv) juggle to apply APOS theory; and (v) create and compile a scenario as shown in Appendix G-2. These important aspects could be done as part of another activity or subject or even added to the previous year of study or grade. This could keep the focus of the learners on Greenfoot instead of login and password management, acquisition of competency in compiling a Greenfoot application, etc.

(b) Analysis and discussions

The task of “how to” login and to ensure that the learners kept track of their passwords and logins took a considerable time. The term “abstraction” was revisited, and learners were asked again to explain the term “abstraction”. This time they had to visit Moodle to assist them in determining how researchers such as Dubinsky (1991), Hazzan (1999, 2003) and Kramer (2007) see abstraction. This also gave learners time to reflect. The “Juggle” schema was tested, and learners could enact “Juggling” since they covered this during a previous intervention.

(c) Findings

Finding 5B-1: Learners used Moodle to seek answers for the basics of Moodle, such as their login and password

Finding 5B-2: Learners managed to find answers on abstraction in Moodle

Finding 5B-3: Learners revisited mathematical expressions with APOS as lens

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Learners took certain steps to visit Moodle and used Moodle as a resource.

(ii) Process

Learners could state exactly what the APOS acronym stands for and they displayed a Process thought process on abstraction and meaning.

(iii) Object

APOS was brought into the context of mathematics, Moodle and programming.

(iv) Schema

APOS became a frame of reference.

(e) Summary

Learners acquired the APOS acronym as a lens when looking at abstraction and mathematics when simplifying fractions. The researcher could see through observation that learners are active in using Moodle and debating abstraction as put forward by Kramer (2007).

**5.4.3.17 Intervention 6: Applying Process and Object within mathematics
(Appendix H)**

(a) Description

The goal of this intervention was to criticise the simplification of an algebraic fraction using APOS as lens. The researcher allowed learners to simplify the algebraic fraction in a natural manner. Terminologies as Action, Process, Object and Schema was used and brought into perspective of the mathematics simplification problem. The researcher wanted the learner to keep an eye on mathematics and APOS theory application in general to avoid learning how to code through rote learning.

(b) Analysis and discussions

Some learners were still at the Action stage. Through observation it was noted that learners immediately started using calculators to divide two into eight. The researcher asked the learners to illustrate how they would simplify this. They still made use of their calculators to divide 8 by 2. Learners did not know their rules on exponents, which indicated that no memorisation happened, hence the manual calculation of division. Being stagnant at an

Action phase will not allow learners to grasp the exponent rules. Exponent rules need memorisation, or it must be looked up every time before it is applied. The researcher made learners memorise APOS and the meaning of the letters of the acronym. The learners understood that the use of calculators was an Action and that no learning took place.

(c) Findings

Finding 6-1: Most learners were still at the Action phase whilst applying simplification, either in using a calculator or having difficulty to apply the rules of exponent usage

Finding 6-2: Learners memorised APOS and understood the meaning of Action and Process from questionnaire answers; the answers provided by these learners show that they made a shift from action to process

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Learners used their calculators to calculate 8 divided by 2.

(ii) Process

Learners had difficulty processing the rules for exponents and did not reach the Process phase.

(iii) Object

No object on this example although exponent mathematical concept was in their curriculum.

(iv) Schema

No schema on this intervention.

(e) Summary

Learners struggled to associate simplification as per the APOS theory. Learners relied on their calculators to perform basic calculations, but their reasoning was limited beyond that. The actual process was not developed yet, as the learners did not think about the mathematical concept definition of simplification.

5.4.3.18 Intervention 7: Greenfoot as process and object (Appendix I)

(a) Description

Learners reverted back to the enhanced GD (Appendix D-2) with flipped classroom techniques. The focus was on the basic creation of a scenario. Learners understood the

Action and Process phases of APOS much better after having completed Intervention 6. In previous interventions they were drilled in writing down the APOS acronym and could achieve the goal. Learners managed to think (reflect) on what Action and Process mean. The goal of this intervention was to create a scenario with APOS theory as lens. APOS theory was focused on in the previous intervention with mathematics as “met-before”. The previous mathematical simplification example provided the learners with more insight into an object in terms of the simplification of fractions and of the Schema, which is an umbrella mathematical concept of simplification. The learners soon realised that their Schema simplification could accommodate basic numbers, but not exponents. The goal of this intervention was to let learners pursue the Greenfoot scenario through APOS theory upon discovering new concepts on programming.

(b) Analysis and discussion

Learners were ready to do coding after creating scenarios earlier. Double clicking on the Wombat Actor opens the editor. From the videos shown in class and on Moodle, learners quickly added the `move(5)` instruction or code, depicted in [Figure 5.34](#). The visual output can be seen in [Figure 5.33](#).

Each time the learner clicked on the run button, the `act()` method executed, continuously, whereas clicking on the act button executed the `act()` method once only. This provided debugging features for learners to measure their coding actions against the outcome. Learners quickly discovered that clicking the “RUN” button executed the `move(5)` iteratively, whilst clicking the “Act” button performed the `move(5)` step at a time.

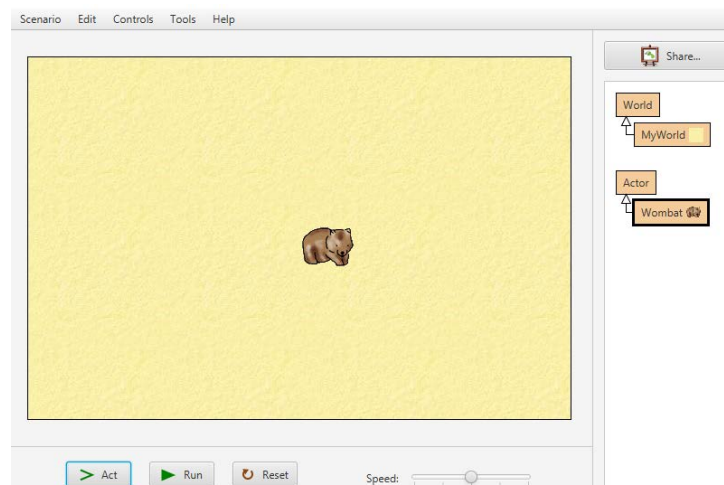


Figure 5.33: The Wombat Actor object within MyWorld World Object

```

Class Edit Tools Options
Wombat x
Compile Undo Cut Copy Paste Find... Close Source Code
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseClick)

/**
 * Write a description of class Wombat here.
 *
 * @author (your name) |
 * @version (a version number or a date)
 */
public class Wombat extends Actor
{
    /**
     * Act - do whatever the Wombat wants to do. This method is called when
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
        move(5);
    }
}
Class compiled - no syntax errors saved

```

Figure 5.34: The Wombat Actor code in Greenfoot editor

(c) Findings

Finding 7-1: Most learners could create a scenario, especially those who watched the videos posted on Moodle

Finding 7-2: Learners understood Action and Process better within the Greenfoot programming language; learners watched the video and then followed suit in creating the scenario

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Learners now understood the importance of an Action in the learning of concept definitions, whether mathematics or programming. They could see how the “Act” and “Run” buttons differ to either enact step-by-step or total execution of the scenario, respectively.

(ii) Process

Learners could transform the Action into a Process, thereby creating another Schema without Moodle or external help. The learners also transformed basic simplification of numbers into a process by not using their calculators upon completion of the coding, after they discovered that they were performing an Action only when they used the calculator.

(iii) Object

The learners realised that there is basic simplification with a number and simplification with exponents, which requires explicit rules that must be followed, and which deals with a different set of mathematical concepts. The learners also saw the creation of the scenario

without any external help as a Process and tried to increase the speed of creating a scenario every time.

(iv) Schema

The “Scenario” creation now forms part of the learner in terms of creating an application Schema. The learner understood compilation before execution.

(e) Summary

The intervention brought both programming in Greenfoot and simplification in mathematics together. Learners discovered that in mathematics and programming, the activities requiring them to learn are the same. They have to go through the action phase first and then these steps, whether in mathematics or programming, are embedded as thought processes.

5.4.3.19 Intervention 8: Rollout of code in Greenfoot (Appendix J)

(a) Description

The goal of the intervention is to ensure that the learner discovers the total Greenfoot framework, with the focus on APOS theory and how it should be done. The intervention was divided firstly into discovering the classes within Greenfoot; secondly, focusing on a method housed within a specific class; finally, rolling out a scenario using the classes and methods. The learners had to find the classes and methods and write them down. This intervention focused on APOS mental structures where the learners discovered the classes and methods.

(b) Analysis and discussions

Greenfoot consists of a fixed number of “classes”. The learners were given a task to discover these classes. They were given opportunity to understand the purpose of each class. During this intervention, the Process stage was obvious, as most learners found it easy to open the Greenfoot programming language and create a scenario with its World and Actors. The researcher pointed out that they know this, as they memorised the Actions and turned them into Processes. Learners actioned the task by watching the videos on Greenfoot scenario and implemented the code in a sequential manner. They also realised they are in the process stage of APOS when the activities were done without watching a video in most cases. Because of the time it took to compile and run a scenario on any terminal, learners complained that they fell short of time. Those who prepared at home and made use of flipped classrooms had no problem in completing the task during class time. The Greenfoot scenario provided immediate output and the response was 100% success or a compilation error. Debugging now also started to play an important role to achieve success. Learners became obsessed in finding a solution.

The Greenfoot Class Documentation option depicted in [Figure 5.35](#) revealed the structure of the Greenfoot programming language to the learners in a breadth-first manner (Zeitz & Spoehr, 1989). They could research the idea of a CLASS and the composition of the Greenfoot package, which consists of classes. The learners also realised that a class consists of methods and hence could find the `isKeyDown()` method within the Greenfoot class and read up on the meaning of such a method. The questions were all answered well by all grade 8 learners, except for one. The learner failed overall because of being absent from the instruction given. The Greenfoot programming language has the help option, as depicted in [Figure 5.3](#). This enabled learners to find the `isKeyDown()` method under the Greenfoot class.

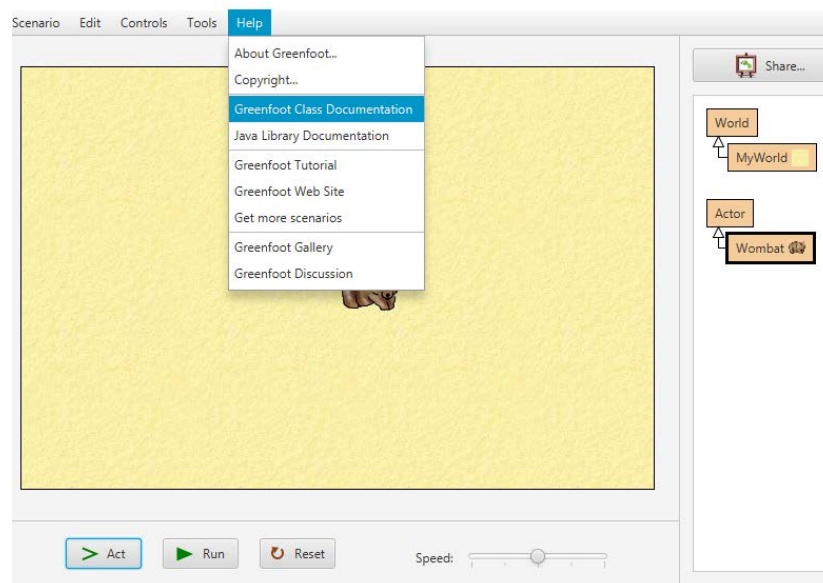


Figure 5.35: Greenfoot Class Documentation

The challenge within the intervention was to solve the dimension representation of a chess board, as well as locating the position of a Turtle object within the World on the chess board, as depicted in [Figure 5.36](#). Although the image of the chessboard consisted of four squares, the dimension of *super* (400, 400, 1) produced the chessboard. The learners soon learnt how to create dimensions and how to determine the x and y positions of any object. This also gave each learner the awareness of position in the World, but it also gave new meaning to the x and y axis in mathematics as a welcome add-on, as depicted in [Figure 5.36](#).

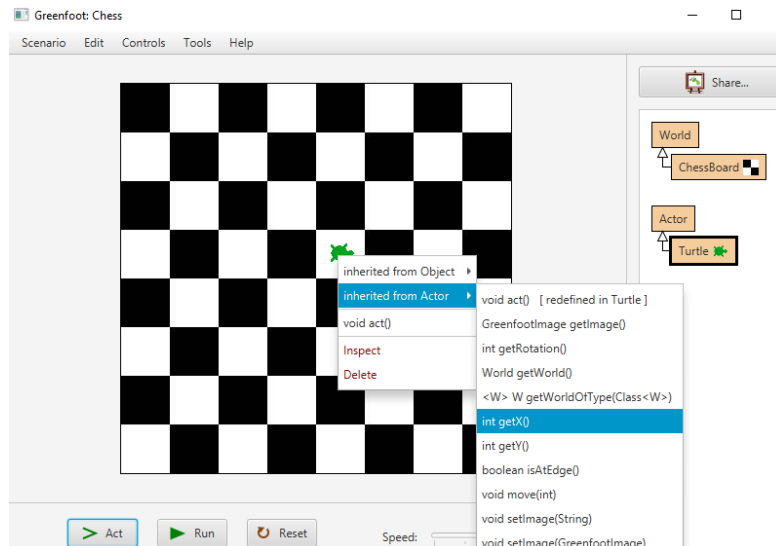


Figure 5.36: Greenfoot ChessBoard World

Based on the instruction given to the learners, they had to create a board of (600, 400, 1), which is 600 pixels wide and 400 pixels on the vertical axis. A student inferred that the output which produces 12 tiles (Figure 5.37), allows each block to be 50 pixels, when looking at his calculations in Figure 5.38. The learner now developed thought processes on how to construct a chess board from the basic square that consists of four squares (Figure 5.36). The four squares alternated black and white tiles. The learner calculated that each block within the square of 4 blocks, consists of 50 pixels. Representing one square with 4 blocks will need a (100, 100, 1) dimension.

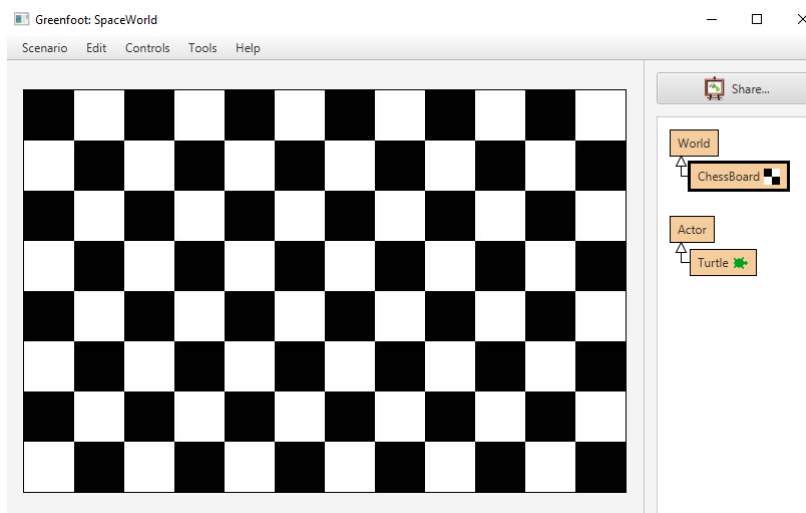


Figure 5.37: Chessboard when using (600, 400, 1) dimension

One student made the following remarks on questions 4.3, 5.1 and 5.2, depicted in Figure 5.38.

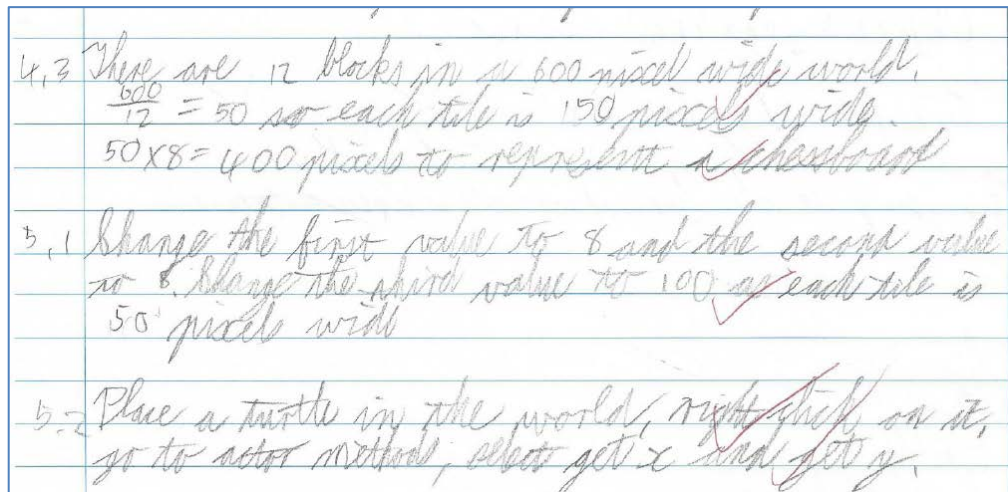


Figure 5.38: Learner calculating the size of the World that will produce a perfect chess board

(c) Findings

- Finding 8-1:** Learners could explain APOS as acronym as indicated by their answers, which emphasised any stage higher than Action in APOS theory
- Finding 8-2:** Learners were able to construct a scenario in Greenfoot, without having to watch a video
- Finding 8-3:** Learners were at least at the Process phase within Greenfoot
- Finding 8-4:** Learners used mathematics calculations in understanding how a chess board was constructed using dimensions
- Finding 8-5:** Learners showed an understanding of dimensions through visualizing the Greenfoot output
- Finding 8-6:** Learners were in a process of finding solutions to problems, indicating thought processes

(d) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Learners now understood the importance of an Action through right-clicking on the Turtle and discovering its x-y coordinate position. Learners could also, through trial and error, set the dimensions and then discover how dimensions are created exactly by inspecting the output after taking action.

(ii) Process

Learners could transform the Action into a Process by guessing the x and y coordinates. The researcher asked them to place the Turtle object on the chessboard world and guessed the x

and y coordinates. The guessing became more accurate the more they memorised the exact locations on account of the dimensions of the chessboard.

(iii) Object

The learners saw methods such as `getX()` and `getY()` and other methods belonging to the Actor class as abstracted representation of their physical actions imposed on the Turtle Actor object.

(iv) Schema

The “Scenario” creation now forms part of the learner in terms of creating an application Schema. The learner could see and understand that the Schema called “Creating a scenario” as a wider application using the assimilation of new concepts into an existing Schema as opposed to accommodation. The researcher also brought this to the attention of the learners. As depicted in [Figure 5.34](#), the learner can either create code such as `move(5)` within the `act()` method of the Actor or select the method to perform the action with a mouse click.

(e) Summary

Overall, learners made progress from the Action phase to the Process and Object phases. Also, new Actions were imposed on the current Schema of “Creating a scenario”, which expanded their Schema on Greenfoot. Through Intervention 8, the learners became more enthusiastic with watching videos and observation. This shows that an understanding of the concepts does help with the motivation of any subject material that needs exploration by learners.

5.4.3.20 Intervention 9: Making decisions towards Encapsulation (Appendix K)

(a) Description

The goal of this intervention was to let learners construct an algorithm to solve the problem of making the Actor sensitive to the edges of the world. The learners need to think about the problem and automate code within a scenario to control the Actor within the world through encapsulation and control structures in Greenfoot.

(b) Analysis and discussion of Intervention 9

Learners were given the challenge to create a scenario and position the turtle Actor object within the world of Intervention 8. They were confronted with the researcher’s application/ scenario in controlling the turtle at the edges of the world. They had to implement Greenfoot code and then relate that to APOS theory. Learners already in the previous intervention experimented with finding the turtle’s location in the world and hence knew about Greenfoot methods such as `getX()` and `getY()` methods through actions. Another important limitation is to conclude any statement with a semicolon, which enforced discipline in having to adhere to

basic syntax. This forced learners to think about the structure of any coding sentence syntactically. The conditional IF statement was needed where learners could position their code within APOS theory. Learners accomplished the decision structure and enacted abstraction such as `atWorldEdge()` as an Object to take Action on.

Learners experienced that a decision was needed by either the programmer or a manual change to be imposed on the scenario. Learners experienced the advantage to have knowledge on methods unlocked by the Process. These methods were discovered earlier and memorised to provide the learners with much more “solution” power than not knowing. Learners also discovered through constructionist learning how to construct code. Debugging also played a role to force the learner to change code and eliminating any typing errors. This forced the learners to focus on correctness.

Learners realised that running a haphazard trial and error process took so much longer than pre-planning the code through creating a proper algorithm. Learners were now confronted with decision structures becoming architects of algorithms. Learners for the first-time encapsulated code and assigned a name to it. The learners then need not worry about the specifics of the code and hence abstraction was done! This forms an Object which learners could relate to and which performs certain functions through lines of code.

Learners were challenged to group/encapsulate code into a method. This is also a form of abstraction by using encapsulation to hide any detailed code. The complex algorithm or sequence of code could now be performed using a descriptive method name such as `atWorldEdge()`. The questions posed to learners, such as “How will the ambulance (Actor object) be controlled through the use of Java code to turn around at the edge of the world?” This question was asked to learners to answer in their mother tongue. All the properties of the problem were highlighted so that they could think about the problem holistically and not simply zooming in on one aspect not related to the whole. See Activities 1 and 2 in Appendix K.

Looking at [Figure 5.39](#), the learners established the sense of width and height of the world the Actor is living in. In this instance, the learner did not understand that the `act()` method would be called iteratively and that the `turn()` method had to make one turn of 359 degrees in an anti-clockwise direction. This is how Greenfoot operates, by calling the `act()` method iteratively and even a small number will turn the Actor during every call of the `act()` method.

Learners were also given graph paper to show the dimension of the world. This further strengthens graphs in mathematics, but it was not a main concern then.

```

if if: (get X() > get World, get Width()
-20) turn (10) (5);
-----
turn (-359);
move (40);

```

Figure 5.39: A learner's response on Activity question

(c) Coding

The Greenfoot scenario depicted in [Figure 5.40](#) and [Figure 5.41](#) reveals the dual-modality of the Greenfoot programming language.

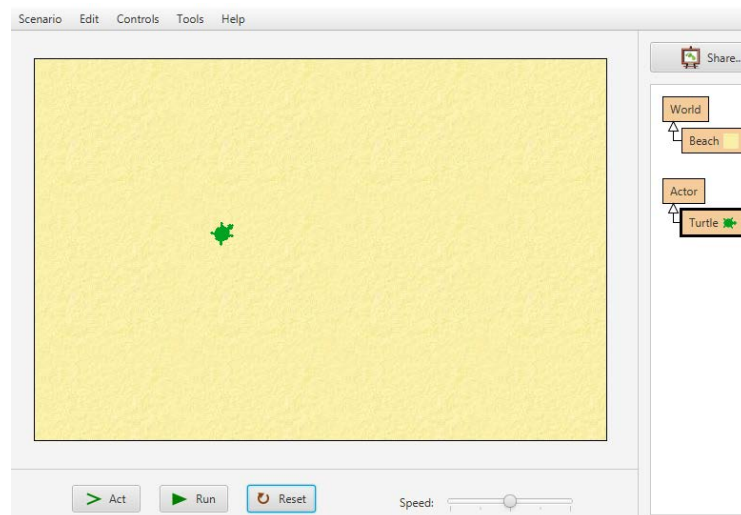


Figure 5.40: TurtleWorld in Greenfoot

The learners were asked to determine coordinates of the Turtle object and in the process they discovered the coordinate map of the World. The learners were then given the code in [Figure 5.41](#) to implement and investigate why the Turtle actually changes direction, although the command is only `turn(10)` degrees. This led to learners to enact the Turtle in the class room, which boils down to an action taken by learners before understanding the command `turn(10)`. The Process changed into an Action.

```

public class Turtle extends Actor
{
    /**
     * Act - do whatever the Turtle wants to do. This method is called
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        // Add your action code here.
        move(10);
        if(getX()<20|| getX()> getWorld().getWidth()-20)
            turn(10);
        if(getY()<20|| getY()> getWorld().getHeight()-20)
            turn(10);
    }
}

```

Figure 5.41: The Turtle act () method coding

Most learners wanted to turn(90) or even turn(359) as indicated by the learner's writing, but realised that repetitive execution of the turn(10) would eventually make the Turtle turn away from the edge. The researcher gave substance to the learners' investigation by directing them to use the Act button and they could visualise the code through Turtle enactment. This is a typical Action taken by the learner to visualise the code. The researcher added to the problem by encapsulating the code into a method atWorldEdge(), depicted in Figure 5.42. The learners now knew that the name atWorldEdge() carried a deeper meaning. The atWorldEdge() is an abstraction of certain commands that achieve one thing in detecting the edge of the World.

```

public void act()
{
    // Add your action code here.
    move(10);
    if (atWorldEdge())
        turn(10);
}

public boolean atWorldEdge()
{
    if(getX() < 20 || getX() > getWorld().getWidth() - 20)
        return true;
    if(getY() < 20 || getY() > getWorld().getHeight() - 20)
        return true;
    else
        return false;
}

```

Figure 5.42: Encapsulating code

Although this seemed very difficult to the average learner, they were able to watch videos, such as Video#5 (<http://wrru.co.za/moodle>) on encapsulation, which explained the process. Every line of code was first enacted and by clicking on the "Act" button the code was executed line by line. This also allowed different abstraction levels of hierarchical design as found becoming an expert by Zeitz and Spoehr (1989). The whole activity helped learners to

see patterns within programming, also called repetitive code, which may be grouped or encapsulated within a method. Any instance was also loaded with properties that promoted the relationship between the learner and the properties of the instance. The tighter the relationship between learner and actor, the better the learners could manipulate the actor instance or object. The relationship of the actor and the learner promotes reflective abstraction. The learners had to enact the actual Actor object as if the learner was projected into the scenario. The APOS theory was brought in context of Greenfoot coding. Reflective abstraction is a huge achievement for learners and their Greenfoot competencies.

(d) Findings

Finding 9-1: Learners understood abstraction through system methods

Finding 9-2: Learners understood embedded code

Finding 9-3: Learners used an IF condition

Finding 9-4: Learners used actions to understand the execution of Greenfoot

Finding 9-5: Learners could enact the problem through trial and error using debugging methods

(e) APOS discussion

The APOS mental structures in terms of the intervention are now discussed.

(i) Actions

Learners enacted the `turn(10)` command using the Act button, which is a higher order than physical enactment.

(ii) Process

Learners accepted the built-in methods of the Actor class such as `getX()` and `getY()` and the Greenfoot class's `getWorld().getHeight()` methods. They need not enact or think about these methods anymore but use them, as they built an understanding through enactment.

(iii) Object

The learners could abstract code through a descriptive name. Although this was a first encounter, most learners agreed that the coding was more legible and they do not have to worry about the code once it performed the steps it was intended for.

(iv) Schema

Learners became excited about new avenues opening to them, which empowered them to do even more in terms of coding. The researcher described this as a Greenfoot programming Schema being expanded with each exploration.

(f) Summary

Learners became aware of encapsulation, abstraction, breaking down an object into an action, discovering computational thinking, based on abstraction and automation.

5.4.3.21 Intervention 10: Revisit encapsulation with the Randomize option (Appendix L)

(a) Description

The goal of this intervention was to let learners impose actions on their current Schema that exists around code encapsulated in a method. This exercise challenged learners to investigate the methods which existed within classes. This exercise was synonymous with the previous exercise, but it added the Randomize function to make the gaming environment more real and unpredictable. The focus is on changing the Schema of the learner, expanding the Schema through assimilation.

(b) Analysis and discussions

Learners found exercise fun, but challenging. Learners now understood abstraction encapsulated within the Randomize method. Learners called and used methods `getWidth()`, `getHeight()`, `getX()` and `getY()`. Gaming became reality owing to random positions. Learners understood that a package consists of classes. The learner manipulated an Actor object to randomly turn in any direction. The learner also quickly realised that accuracy when using method names were of the essence and a sense of perfectionism was instilled in every learner or else there was no success. After each compilation of the Greenfoot code, the learner was confronted with debugging exercises.

In mathematics, such problems were left to the teacher to either mark it right or wrong, but programming allowed the learner to immediately witness whether the outcome was correct, by forcing the learner to apply corrections after each compilation. The researcher regards patience as a competency when stuck on a problem and persevere until it is solved.

(c) Coding

The learners were requested to do research and found the application of the Randomize method within the Greenfoot class. Learners were quite relaxed in extracting information on the method from the Greenfoot API and seemed quite efficient. Randomization was used to make the flow of the scenario more natural.

```
getRandomNumber  
public static int getRandomNumber(int limit)
```

- Return a random number between 0 (inclusive) and limit (exclusive)

- Parameters:
- limit – An upper limit which the returned random number will be smaller than
- Returns:
- A random number within 0 to (limit-1) range

The learners could understand that 0 was inclusive, but the number or upper limit excluded. The turn command will therefore receive a value of either 0 up to 99 when using the following code snippet depicted in [Figure 5.43](#).

```
public void act()
{
    // Add your action code here.
    move(10);
    if (atWorldEdge())
        turn(Greenfoot.getRandomNumber(100));
}
```

Figure 5.43: Greenfoot code illustrating the “turn” command

(d) Findings

Finding 10-1: Learners automate the code in natural a way

Finding 10-2: Learners discovered the Randomize method through investigating the Greenfoot classes

Finding 10-3: Learners investigated the rules of the Randomize method from help documentation as done during intervention 2B, before implementing

Finding 10-4: Learners could connect abstraction to built-in methods and self-declared methods

(e) APOS discussion

(i) Actions

Learners interacted with the Greenfoot class methods within the Greenfoot Class Documentation under the Help menu option. Intervention 2B gave learners a “met-before” in how to use the online documentation options.

(ii) Process

Once the learners investigated the `getRandomNumber` method, the implementation allowed them to memorise the meaning of the method. Other methods were also seen during their investigation, which broadened their knowledge. They could then just apply the method and associated methods.

(iii) Object

Learners became aware of more methods within the Greenfoot class, which became Objects towards their coding.

(iv) Schema

Learners became much more fluent in Greenfoot programming and expanded their Schema by imposing Actions on the Object of a Greenfoot scenario.

5.4.3.22 Intervention 11: Assessment (Appendix M)

5.4.3.23 Intervention 11A: Informing the learners of the assessment in a structured manner (Appendix M-1)

(a) Description of Intervention

The goal of the intervention was to guide the learners with preparing for an assessment. The intervention focused on informing the learner in a structured manner of what they were writing on. This minimised anxiety.

(b) Analysis and Discussion of Intervention

Learners found the guide on preparation extremely useful, which minimised their anxiety.

Learners understood what was being assessed. The learners were prepared for the assessment using Greenfoot and Moodle as resources. Learners now used Greenfoot as a reliable resource with Oracle prepared notes and videos by Prof Michael Kölling (Greenfoot, Code, 2014). All exercises and pre-tests or assessments were available to the learner. The teacher could also verify if the learner logged into the system to prepare for the assessment. Resources such as Moodle were also tried and tested. Learners were prepared for the assessment through practical exercises and were looking forward to the assessment.

(c) Findings

Finding 11A-1: Learners appreciated the tangible breakdown of what to study for the assessment

Finding 11A-2: Learners consulted Moodle as source of information

(d) APOS discussion

(i) Actions

Learners were informed of the context of the test and could take action in preparing for the assessment.

(ii) Process

Learners memorised most of the code through implementation and research.

(iii) Object

Learners regarded assessment as a measure of their competencies in the Greenfoot programming language.

(iv) Schema

Learners could visualise the assessment to assess their programming competency and understood exactly what was expected of them.

5.4.3.24 Intervention 11B: Assessment in Greenfoot on Encapsulation and problem solving (Appendix M-2)

(a) Description of Intervention

The goal of the intervention was to let learners solve a problem using Greenfoot with code as output. Learners developed a scenario from scratch by creating their own Actor images and background for the World class using Paint. The aim was to control movement in Greenfoot. Learners had to understand how the world is constructed in Greenfoot. Learners constructed a World and Actor objects using Paint. They then put together a game with their own constructions and applied code through encapsulation (OBJECT).

(b) Analysis and Discussion on Intervention

Learners enjoyed the assessment thoroughly during the observation and capturing on video.

Learners could devise a solution/algorithm to integrate the Paint object into a scenario. Learners understood the complexities of encapsulation through construction. Some learners could create structures in Greenfoot which pointed to the existence of a Schema. Learners manipulated sound and their own Actor and World objects beyond the Greenfoot given World and Actor objects. Learners discovered their creative side through using Paint and incorporating the Paint object in Greenfoot. This was then loaded as a World object as background or backdrop to the game they devised. The learner was tested to his/her limits. The learner created his/her own background for a World and in this case, a racing track populated by trees and other objects. The learner then had to control actors such as a car or vehicle with the cursor keys of the keyboard and guide the vehicle all along the road to the finish point. Obstacles along the road alerted the learner if the vehicle actor instance collided with objects such as trees or houses. The same coding was examined in the Crab scenario, available in the Book Scenarios and in the videos. Here the Lobsters ate the Crabs, which illustrated what happened when Actor objects invaded one another's space and the action that had to be taken through calling methods.

The learner was forced to visit the Moodle site and watch the video in preparing for the activity.

The researcher also created videos where the learners worked in groups of two, and the video illustrated the motivation and keenness of the learner to accomplish the goal of creating the game. The activity was thoroughly enjoyed by both learners within each group. Learners constructed the most fantastic designs and asked quality questions on “how to” make the Paint jpeg file part of the Greenfoot scenario. This forced learners to explore the folder structure of Greenfoot and where these files should be stored. This was covered when they started out with Greenfoot and they had to unlock that part of their programming experience. Some also added sound images to the car and sounds simulating a crash when the vehicle collided with an object, thus making the game so much more interesting.

Aspects such as encapsulation, if-statements, understanding the dimensions of the world, exploring the properties of actor instances in the scenario and coding in general were constructed. The constructionist approach rested upon tried and tested theory within Moodle. The APOS theory was re-enforced and the learner was asked to identify APOS elements within the Paint exercise. A small number of learners identified and created an OBJECT and a SCHEMA construction within the exercise. The researcher assisted to identify custom methods as objects within the scenario and showed them how to move repeating code into a method and just calling a method, not repeating the code, as illustrated in the following [Figure 5.44](#).

```
public void lookForTree()
{
    if ( canSee(Tree.class) )
    {
        collide(Tree.class);
        Greenfoot.playSound("su.wav");
        Greenfoot.stop();
    }
}
public boolean canSee(Class cls)
{
    Actor actor = getOneObjectAtOffset(0, 0, cls);
    return actor != null;
}
public void collide(Class cls)
{
    Actor actor = getOneObjectAtOffset(0, 0, cls);
    if(actor != null) {
        getWorld().removeObject(actor);
    }
}
```

Figure 5.44: Greenfoot code for repeating code

(c) Findings

Finding 11B-1: Learners incorporated other tools such as Paint to construct a background or new World

Finding 11B-2: Learners applied encapsulation by moving code into a method

Finding 11B-3: Learners involved control structures into their code

(d) APOS discussion

(i) Actions

Learners watched videos on how to create an image in Paint for a background.

(ii) Process

Learners knew the folder structure of Greenfoot and immediately saved the background in the images folder.

(iii) Object

Learners saw the Greenfoot programming language scenario as an Object to construct World and Actor objects.

(iv) Schema

Learners added many more methods to their Greenfoot Schema, which expanded into lines of code that constitute a scenario.

5.4.3.25 Intervention 12: The Variable in Greenfoot (Appendix N)

(a) Description of Intervention

The goal of the intervention was to introduce learners to a basic variable and show how to use Pascal or Camel case when defining these variables. Previously, no emphasis was placed on the exact purpose of a variable. Now the focus is on describing a variable properly and combining the variable as part of an IF-statement. Learners had to understand a basic variable in Greenfoot. They had to apply a variable within code, understand how a controlling statement is used, understand a Boolean variable, and had to investigate the proper notation of a variable.

(b) Analysis and Discussion of Intervention

The activity was difficult, as the learners had to understand how an exclamation mark or negation character describes the negative of the current situation. This was easily explained and illustrated – when a rocket moves in a specific direction and when it touches the opposite side or another object the direction is negated, which then gives a response to driving the rocket in the opposite direction by subtracting from the x or y value. The learners experienced a Boolean variable controlling the rocket movement and direction. Learners had difficulty with the syntax at first, but after witnessing the output in having an “!” in front of the Boolean variable taking an action to subtract from the x or y value, brought clarity to their understanding, such as: `if (!forward) xValue = xValue – 5` and so on.

(c) Challenges

The problem assigned to the learners was to create an object anywhere in the world. They then had to make the object move up and down, depending on the direction and controlling Boolean variable. At this stage, the coding was becoming challenging if learners missed previous sessions or did not complete exercises. The academic school year also came to an end and teachers and learners focused on studying for their assessments. This distracted them from using Moodle or devoting as much attention as they have done throughout the year. This made their understanding and interpretation very difficult in terms of find an algorithm for a solution for the problem at hand. Moodle was always there for them to utilise, but it was the researcher's experience that the process of enforcing the learner to visit the Moodle site and "pick up the pieces" would only realise if the subject was slotted into the school system as an examinable subject. Due to some of those elements being missing, the students returned unprepared, which created frustration within the learners and in the researcher. Learners without home PCs even made the situation worse. As part of the GD, two videos were created as flipped classroom techniques and an introduction for the learners depicted in the "coding" section below. The learners could now create their own conceptual image of the video that portrayed the conceptual definition of the problem.

(d) Findings

Finding 12-1: Learners could understand the usage of a Boolean variable as a controlling variable, but with challenges

Finding 12-2: Learners applied the `getX()` and `getY()` methods

Finding 12-3: Learners understood the dimensions in terms of x and y values that make up the grid

Finding 12-4: Learners did not prepare as well, or use Moodle as often as before, because the subject was not a 'real subject' that influenced their future

Finding 12-5: Learners found the coding challenging when they had to apply syntax

(e) APOS discussion

(i) Actions

Learners engaged with `getX()` and `getY()` methods by clicking on `act()` method and experimented with the moving of the object in the world.

(ii) Process

Learners used the IF control statement to determine the direction of the object.

(iii) Object

Learners could visualise the movement of an object in a vertical position.

(iv) Schema

Learners added many more methods to their Greenfoot Schema, which expanded into lines of code that constituted a more complex scenario.

(f) Coding

To illustrate how the variables dovetail into a game, the “worked-example” effect was rolled out to as two videos to the learners, which can be viewed at link:

<http://www.wrru.co.za/moodle/mod/resource/view.php?id=542>

and

<http://www.wrru.co.za/moodle/mod/resource/view.php?id=543>

The two videos use breadth-first hierarchical organisation for the learners’ understanding.

5.4.3.26 Intervention 13: Moving from Process to Object in APOS using Greenfoot (Appendix O)

(a) Description of Intervention

The goal of the intervention was to change the scenario of the Greenfoot application into a turnkey application. Each learner gained more information and insight into the Greenfoot environment through the Greenfoot application using the Moodle LMS as aid to assist in pre-preparation. The learners were challenged to make the scenario a turnkey application. The learner had to position the Actor object when the scenario opens, within the World at a specific position. The scenario portrayed red and blue balloons, which changed position as scenario executed.

(b) Analysis and Discussion on Intervention

The APOS acronym formed part of the learners’ arguments when dealing with Greenfoot. Learners understood Greenfoot vocabulary within editor-compiler and help files. Learners married Greenfoot as learning goal and Moodle as resource to explore and find solutions or answers to their problems in Greenfoot. Learners understood the graphing of x and y axis values owing to the constructionist approach in order to witness the outcome of their actions through enactments in Greenfoot using the act() method. Although this was not the aim to learn mathematics, the x and y coordinate concept became reality owing to enactment, when the learner became the Actor (Balloon) in the scenario.

A mark out of 20 was assigned for each attempt, and most learners achieved above 80% overall. Fourteen (14) English speaking learners completed the assessment and obtained an average of 91%, whereas eighteen (18) Afrikaans speaking learners completed the assessment with an average of 88%. After exposing the learners to the preparation modules, i.e. Moodle and the Greenfoot application, the learners now researched the information on

the Internet and Moodle LMS. They were then subjected to a class test using the Greenfoot editor and compiler as resources to answer the questions above in Appendix O.

The aim of the assessment was to verify that the learners understood the dimension structure of the world within Greenfoot and the commands needed to manipulate an actor within the world in all four directions – north, east, south and west. This affected the Action, Process and Object stages of APOS theory.

Ten English speaking learners and seventeen (17) Afrikaans speaking learners participated in the assessment.

Questions 1, 2 and 3 were answered quite well. Except for one student who struggled to create a scenario, and compile and add actors to the scenario, everyone obtained full marks for these questions. The learners understood the dimensions of the world regarding the x and y coordinates and indicated correctly the directions by which the x and y values increased and decreased using the 0,0 coordinate in the left top position of the screen. Attention was required for question 4 and question 5, which needed understanding of the movement of a red and blue balloon along the y-axis and the x-axis respectively. The learners also discovered and understood that the `act()` method is called every time the scenario is executed or run. The learners used the `setLocation(x,y)` command, and by doing this, they had to change the x value for the blue balloon and the y-value for the red balloon.

Three (3) out of the 10 learners and 8 out of the 17 learners answered question 5 very well. This demonstrated their insight into the actual location and movement of the red and blue balloon actor instances. The Afrikaans speaking learners averaged 79% and the English speaking learners averaged 76% for the assessment. The learners showed a strong enactment with the actors in the world, which guaranteed the correct x-y coordinate association of a balloon in the world.

The learners also grasped a very confusing and difficult concept in that of $y=y-5$ whereby the y coordinate was incremented each time the `act()` method was called. When looking at this equation at first, it simply does not make sense that $y=y-5$. Question 3 still demanded physical action by the learners in most cases. Question 4 and question 5 indicated that the Process stage of APOS made sense to most of the learners in that they started to visualise the exact location of the balloons without taking action. Taking action means that the learners would have used `inspect` or the same method to show the coordinates of the balloons wherever they were positioned as opposed to visualise and memorise the world's dimensions in terms of x and y coordinates. Twenty percent (20%) of the learners still dragged the red and blue balloon to a location as per stated problem and then inspected the

x and y values for that object, which is better than guessing. In this way, they discovered the positions and whether x or y incremented. This also gave them an understanding of the graph coordinates. Question 4 and question 5 demanded more from the learner in terms of utilising his/her memory to represent the balloon at a location visualised in the learner's mind. This also implicated a process within APOS, which could only have developed if the learner enacted the action into a Process and did not rely on an action to understand question 4 and question 5.

The learners made use of the Moodle LMS and discovered the theory of Greenfoot as part of a constructionist learning activity. Students achieved the basic minimum requirements of being able to log into Moodle and retrieve information to answer the questions, with login details as provided in the task. The researcher noticed that some of the introvert learners were also beginning to ask questions in class, making them heard. The reason attributed to this is that those who wanted to do so, also downloaded the source code from the Moodle site and installed Greenfoot on their home PCs. This enabled accelerated learning at school during the sessions.

Overall, most learners could write down the APOS acronym and explained what it stands for. They also understood the Greenfoot language as consisting of classes and knew exactly where to click to show the documentation within the editor-compiler. The learners understood what was meant by dimension of the chess board and they could position an instance of an actor at a specific x-y location. They understood that a class consists of methods and that an instance of an actor class may participate in the specific world per definition. In summary, the learners knew that they could fall back on the Moodle repository to find answers to the questions, with results depicted in [Table 5.10](#).

Table 5.10: Test 2 results

Test 2	Total Written	Average %
English	10	76
Afrikaans	17	79
Total Learners	27	77

(c) Coding

The learner identified `setLocation(x,y)` where x or y changed to satisfy the goal of advancing the red or blue balloon. The movement process was governed by a Boolean variable called "up". The "up" variable was used in conjunction with the red balloon going up or down.

(d) Findings

Finding 13-1: Learners used an Action to gain clarity on the changing x and y values, which was achieved by physically dragging the red or blue balloon to a position on the designated path and inspect the x and y values

Finding 13-2: Learners experienced challenges when they jumped straight into a Process, instead of taking Action. The coding then became a guessing exercise

Finding 13-3: Learners understood the essence of a Boolean variable

(e) APOS discussion

(i) Actions

Learners took Action by dragging the red and blue balloons to a location as per instruction.

(ii) Process

Learners could use variables to imitate the balloon movement.

(iii) Object

Learners acquired “movement” as an Object in manipulating x and y settings.

(iv) Schema

The Schema expanded in accommodating variables as part of the Greenfoot Schema.

5.4.3.27 Intervention 14: GD creation on IF statement

The goal of Intervention 14 was to create a GD on the IF statement and on peripheral programming language concepts. In the New Year, the English speaking group could not be slotted into the timetable, thus only the Afrikaans speaking group of 2015 carried on. The researcher decided to ‘throw’ problems at the learners all the time and assess if APOS thoughts were applied when solving these problems. The GD should then allow learners to comprehend the IF concept in programming, which was re-applied after the sub-interventions were completed. These sub-interventions contributed to the GD. In order to get to a GD on the IF statement, the intervention was subdivided into several sub-interventions as stated below. The GD would then be perfected and tested again to ensure correctness, validity and reliability. The GD consists of activities, class discussions and exercises (ACE) as depicted in Appendix D-1.

Intervention 14A: Basic understanding of a scenario with World and Actor classes

Intervention 14B: Manipulation of Actors in a World

Intervention 14C: Interaction of Actor within the world solving problems

Intervention 14D: Adding graph paper and Greenfoot to develop an algorithm

Intervention 14E: The IF statement as a solution to address problems

5.4.3.28 Intervention 14A: Basic understanding of a scenario with World and Actor classes (Appendix P-1)

(a) Description of Intervention

The goal of this sub-intervention was to determine the learners' understanding of the basic concepts of a scenario. The new school year started, and the researcher had to refresh learners' perspective of Greenfoot programming. The understanding of basic terminologies such as scenario, World and Actor classes were assessed. Learners could refer to the GD developed in Appendix D-1 and D-2.

The activity was to use Greenfoot programming language and create a scenario called ArabianNights.

The class discussions were held through learner-researcher interaction or learner-on-learner interaction. The exercises in this case were the problem to explain in words what learners understood by the ArabianNights scenario. What had to be embedded in the mind before attempting the scenario.

(b) Analysis and Discussion of Intervention

Learners had to familiarise themselves with the basic concepts. All these concepts, such as class, object, world, Actor, compile and usage of Moodle, were enforced. The questions were then given as part of this intervention, to test the learner's response. Learners did not access Moodle, as their responses were that they have already done so the previous year. Question 7, [Figure 5.45](#), was answered well in that the learners already transformed the Actions as thought processes in their minds to accomplish the task at hand. The Process phase was therefor already entered, according to the APOS theory. They created the scenario ArabianNights, being able to refer to the first GD namely "Creating a scenario" (Appendix D-1, intervention 2A). Learners found it easy to create the trees along the pathways the object would travel.

[Figure 5.45](#) illustrates a typical answer that portrays a lot of detail connecting to physical steps.

7. Voor jy die ArabianNights scenario geskep het, wat het deur jou brein geflits om dit te kan skep (stappe); dit is, hoe visualiseer (beeld in brein) jy die proses of sien jy dit as 'n volgorde van aksies hoe om die scenario te voltooi. Beskryf die beeld in jou brein om die scenario te voltooi.

- Ek het op subclass gegaan en die class ~~akte~~ class dessert genoem en 'n sand.png gekies (en dan compile)
- Toe het ek by die class Actor 'n objek gemaak deur New subclass te klik en toe noem ek dit Camel en kies 'n Camel.png.

Huiswerk: Kyk asseblief videos 4 en 5 voor jou volgende klas. Skryf in huiswerkboek asseblief.

- Ek doen toe weer dieselfde en noem dit toe Appte AppleTree en kies toe 'n tree.png en toe compile ek dit. Hoe
- Toe add ek 'n objek (Camel en AppleTree) 2

Figure 5.45: Question 7 answer

Question 7: Before you created the ArabianNights scenario, what flashed through your brain to create the scenario (the steps). That is, how did you visualise the process, or did you see it as a sequence of actions to complete the creation of a scenario? Describe the image you have within your mind to complete the scenario.

"I went to the subclass and called the class Desert and chosen a sand.png (and then compiled) – Then I went to the Class Actor and made an object of the Actor class, by clicking on new subclass and I called it Camel and chosen a camel.png."

Homework: Please watch videos 4 and 5 before you attend the next class. Write this down in your homework book please.

"I did the same and called it AppleTree and chosen a tree.png and then I compiled it. Then I added an object (Camel and AppleTree)."

This shows that a Process phase of APOS theory was entered into by the learner. The learner knew exactly what to do and could describe every detail in accomplishing the task in solving the problem.

(c) Findings

Finding 14A-1: Some learners had to make use of the first GD as actionable steps

Finding 14A-2: A Process (APOS) was present, especially in answers provided for question 7

Finding 14A-3: Learners were not keen to watch videos again once they have been there, and could skip some steps in the GD

Finding 14A-4: Learners showed that a Schema did exist, which contributed towards providing a solution to the problem

(d) APOS discussion

(i) Actions

Learners did not watch videos and the Action phase was in background. Some learners referenced the GD, which points to an Action phase.

(ii) Process

Learners could describe the Actions embedded within their minds, which pointed to the Process phase.

(iii) Object

Learners knew exactly what to do and brought back the Object of “Creating a scenario” captured in GD in Appendix D-2, which formed part of their schema.

(iv) Schema

The Schema was recalled and updated.

5.4.3.29 Intervention 14B: Manipulation of Actors in a World (Appendix P-2)

(a) Description of Intervention

The goal of this sub-intervention was to create a scenario called Moon and control a rocket within the Space world and guess its position in the world. The difference when comparing this intervention with the previous one is found in augmenting code and understanding the world dimension. That is, the movement from drag and drop to coding. Actions triggered in the mind when receiving the challenge or problem, were highlighted. Learners were asked to use Greenfoot and create the World and Actor classes, after refreshing the Greenfoot IDE during previous lessons. They were then asked to explain what they have done by writing down their thoughts. A simple `move()` method was used to move the Actor rocket object.

(b) Analysis and Discussion of Intervention

The learners enjoyed the activity. They have done this before and could relate to the GD in Appendix D-2 and recalled their Schema on Greenfoot that came from the previous intervention. From observation, the learners were eager to participate, and the problem given was accomplished with ease. However, from the answers received such as “**What is the goal of the Compile button?**” learners were not knowledgeable on the compile concept. As depicted in [Figure 5.46](#) it shows that an answer given by a learner was vague. The learner used words such as “all or everything that were done must be created”. This is true to a certain extent, but still, the words are generic and not specifically pointing to the concept.

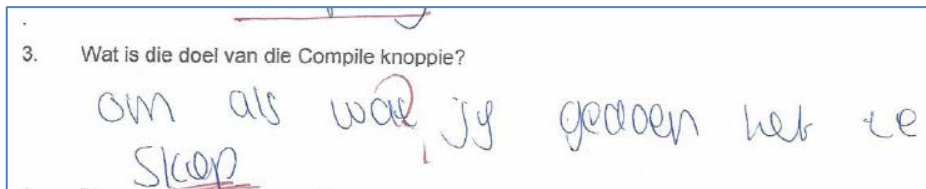


Figure 5.46: Question 3 on the term 'compilation'

The best answer was provided in [Figure 5.47](#) (Translation: "To ensure that one does not make any mistakes in one's programming and that starts/runs the scenario") where the learner stated that it is about the removal of syntax errors and to ensure the scenario is started.

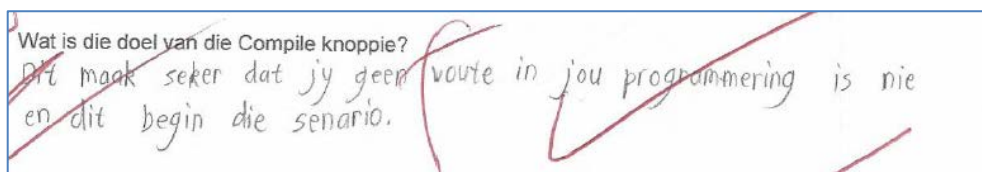


Figure 5.47: Answer to what 'compilation' stand for

They were ready to control their objects using the IF statement. The questions were answered in a much more professional or scientific manner, which indicated that learners are in control.

(c) Findings

Finding 14B-1: Learners achieved the creation of World and Actor classes as a Process

Finding 14B-2: Learners could also manipulate the rocket Actor object in turning it 90 degrees

Finding 14B-3: Learners were unsure about the location of the rocket after the movement

(d) APOS discussion

(i) Actions

Learners did not watch videos and the Action phase was in background.

(ii) Process

Learners could describe the Actions that were in Process phase, with ease.

(iii) Object

Learners knew exactly what to do and brought back the Object of "Creating a scenario", which formed part of their schema.

(iv) Schema

The Schema was recalled and augmented.

5.4.3.30 Intervention 14C: Interaction of Actor within the world solving problems (IF statement as precursor to GD) (Appendix P-3)

(a) Description of Intervention 14C on 06-03-2015

The goal of this intervention was to let learners solve the problem of how to control the object when reaching the end of the world. A hypothetical genetic decomposition (GD) is also a by-product of the outcome of the activity. Learners had to make choices to control the object (ambulance or turtle – learners were given free choice on what object to choose). The task consisted of 4 activities or class exercises. The focus of the activities was on investigating the IF statement. Methods that output the location were used. The learners were given a refresher tutorial before attempting this test as part of the class activities, discussions and exercises (ACE). The learners were introduced to the IF statement. Prior to this test, the learners were constantly reminded to watch the videos on Moodle and do a tutorial to guide them for this test/task. They had to indicate which videos they watched, and their answers reflected clearly when videos were not watched. The researcher brought it to their attention.

The questions focused on the problem of how the object can be guided to turn at the edges. This is typical of computational thinking – in posing a problem to the learner and the learner must provide a solution considering a number of variables, i.e. location of the object relative to the x and y axis, Boolean variable use, and control structures such as the IF statement became the focus. As indicated in Figure 5.48, the learner understood embedded methods such as `getX()` and `getY()` and `getWorld.getWidth()`.

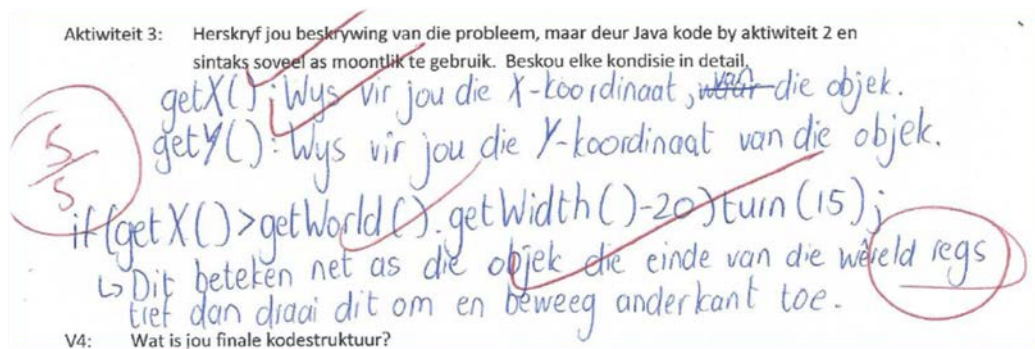


Figure 5.48: Answer consists of Greenfoot code for checking edges

Translation:

Activity: Rewrite your description of the problem, but use Java(Greenfoot) code for activity 2 and use syntax as much as possible. Consider every condition in detail.

The answer portrayed in Figure 5.48 clearly shows that the learner understands abstracted methods in terms of the x and y coordinates. In this specific case, the `getWorld().getWidth()`

method was used, where with previous answers, the learner looked up the dimensions of the world to establish the last x coordinate position, depicted in Figure 5.49.

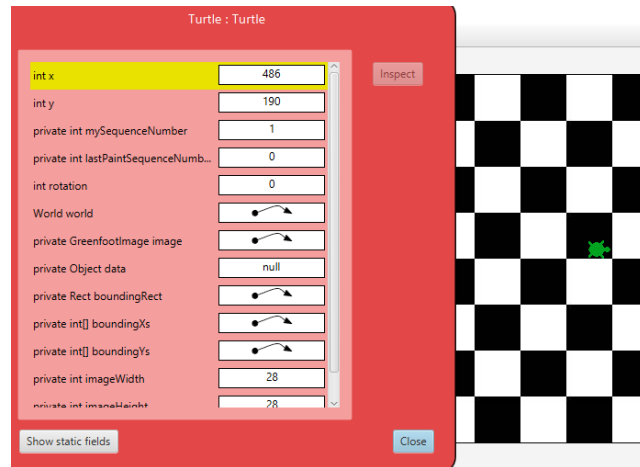


Figure 5.49: Inspection parameters on Turtle object

The researcher also handed out graph paper so the learners could relate to the x and y coordinates. The learners soon realised that these coordinates could be inspected onscreen and that they did not need the graph paper pages.

(b) Analysis and Discussion of Intervention

The learners gave good answers in terms of detailed descriptions of what should happen when the car or turtle reached the edge of the World.

VRAAG 1: Wanneer moet die motor op horisontaal omdraai? Beskou elke geval. Skryf die kode neer wat jy sal gebruik. Doen elke kant apart. Moenie c doen alvorens jy a en b voltooi het nie.

Wanneer die motor die einde van die wereld bereik dan moet die motor omdraai. (in my geval 40 minder as einde van wereld) Wanneer dit by regterkant gedraai het en by die linker kant kom dan draai dit weer ps dit by die einde van die linker kant kom.

c. Jou motor beweeg nou net horisontaal. Jy wil nou die motor in enige rigting laat beweeg en ook wegbeweeg van bo en onderkante van die world.

Bepal eers deur te bepaal wanneer die motor by bo en onderste kante van world kan wees en gebruik die "if" stelling om te verhoed dat die motor links, regs, bo of onder gaan bots, maar wegdraai. Soek metode getRandomNumber(180) op in Help se Greenfoot Class Documentation.

(in my geval is die einde 40 nader)

Figure 5.50: Answer to Question 1

Translation:

Question 1: When should the car turnaround on the horizontal? Consider every case. Write code you will use. Do each side separately. Do not number c before completing a and b.

- c. Your car now moves horizontally. You want to allow the car to move in any direction and also move away from the top and bottom edges of the world.

Firstly, plan by determining when the car will reach the top and bottom edges by using an IF statement to prevent the car from colliding with the left, right, top or bottom edges. The car must turn away from the edges before a collision takes place. Look up method getRandomNumber(180) by using the HELP option in the Greenfoot Class Documentation.

The student wrote in [Figure 5.50](#): “When the car reaches the end of the World, it should turn around (in my case, it should turn around 40 steps less than the end of the World). When reaching the right side, it must turn again (in my case, the end is 40 steps closer)”. This typical thinking shows that the student is conveying the message in algorithmic terms, being very specific. The learner understood the precise steps that had to be coded to ensure a solution to the problem. The learner however saw no built-in abstraction methods that could have assisted. The answer illustrates that the learner’s thought processes are structured, but in a manual way keeping to the basic steps. No use of abstraction is shown in the answer, such as the use of system methods.

Another student showed a different, but more in depth understanding of what the code entails, as depicted in [Figure 5.51](#).

VRAAG 1: Wanneer moet die motor op horisontaal omdraai? Beskou elke geval. Skryf die kode neer wat jy sal gebruik. Doen elke kant apart. Moenie c doen alvorens jy a en b voltooi het nie. (10)

Dit moet beweeg en as dit naby die rand kom, as die x-koordinaat in 'n sekere marge kom moet dit in 'n lugrake rigting beweeg. draai.

```

move(5); // om dit te laat beweeg
if (getX() > getWorld().getWidth() - 5) // Draai aan regterkant.
    turn(Greenfoot.getRandomNumber(180));
if (getX() < 5) // Draai aan linkerkant.
    turn(Greenfoot.getRandomNumber(180));

```

Jou motor beweeg nou net horisontaal. Jy wil nou die motor in enige rigting laat beweeg en ook wegbeweeg van bo en onderkante van die world. 'n mens kan 'n random turn gebruik om dit in verskillende rigting laat beweeg, maar as die y-koordinaat in 'n sekere marge kom dan moet dit draai.

VRAAG 2: Wanneer moet motor van bo en onderkante wegdraai? Beskou elke geval apart. Skryf die kode neer wat jy sal gebruik. Doen elke kant apart. (10)

As die y-koordinaat minder as 5 is moet dit in 'n lugrake rigting beweeg en as dit groter as 395 is moet dit in 'n lugrake rigting draai.

Figure 5.51: Response from student regarding Car movement at edge of the World

The translation of **Question 1** is the same as in [Figure 5.50](#). [Figure 5.51](#) illustrates a different understanding or angle to the problem. The learner referred to the x and y coordinates that needed monitoring. The learner also made use of a random number generator and the learner embedded that into the turn(Greenfoot.getRandomNumber(180)) method. Much

more intelligence was built into this learner's thoughts, as he/she tracked the x and y coordinates, using `getX()` and `getY()` built-in methods. The answers to these methods were compared to the actual width and height of the World, instead of hard coding those values.

(c) Coding

Learners struggled with the task in general because many did not watch the videos that were prescribed for the task. Some learners described the problem by breaking it down into steps and assigning code to each step. Some learners could not describe the problem in general but only focused on a specific side, i.e., right or left and forgot about above and below. Some even understood the term inheritance and it became obvious that they could write code to detect every point within the dimension but had to generate general code driven by a variable. Some students realised that by having a `move(x)` as the first command within the `Act()` method, the object may be against the edge and will stay there forever, hence the coordinates should be obtained for the object prior to movement taking place and only advance the object if there is space or turn the object if the edge was reached.

The following code only tests for the right-hand side of the World.

```
public void act()
{
    // Add your action code here.
    move(30);
    if (getX() > getWorld().getWidth() - 20) turn(20);
}
```

This is typical of a learner that entered an object phase seeing the IF statement as part of a generalised structure. Others did not describe the process in words, but used Greenfoot as computational notation to code with a trial and error approach within the `Act()` method. Overall, with exception of a few, learners have not as yet made an object from the processes they knew, i.e., create an additional method to be called first before moving the object further. Abstraction played a prominent role in this activity, which could be realised through inheritance of a common method. One learner pointed to inheritance as the major factor in the problem.

As part of preparation for the test, some learners returned to the Action phase to gain clarity on the problem. They had to walk around in the classroom with his/her eyes closed and the fellow learners communicated commands to guide the learner so as not to bump into objects or the wall. This showed that learners remembered the APOS theory. The researcher also re-emphasised the APOS theory in the video that was produced to discuss the IF statement

within the Greenfoot code. This gave each learner a sense of certain steps and commands that needed to precede others to make the journey successful.

(d) Findings

Finding 14C-1: Learners had different Schemas on Greenfoot in terms of expertise

Finding 14C-2: Learners described the problem better in context of coding

Finding 14C-3: Learners enacted the “Car” object while walking around blind-folded in the class guided by their fellow learners, i.e. physical steps and movements of the Car and the classroom being the edge

(e) APOS discussion

(i) Action

Learners still enacted the trajectory of the Car object by walking in class simulation the Car’s movement in an attempt to resolve the problem.

(ii) Process

Some learners were writing and describing the solution in much detail, without enacting.

(iii) Object

The Object on the IF statement took on form as a control structure concept.

(iv) Schema

The Schema was expanded again when some learners introduced more methods to automate and code in an abstracted manner, by using pre-existing methods. The majority still struggled with the IF statement.

5.4.3.31 Intervention 14D: Adding graph paper as part of GD to develop algorithm (Appendix P-4)

(a) Description of Intervention

The goal of the intervention was to link APOS and augment the GD of the IF statement with the learners’ coding strategy. The intervention wanted the learners to discover where the object could fall off the World. The learners encountered problems with the IF statement during previous test. Graph paper was now added as part of the test to represent the object as per scale compared to the dimension. In the previous scenario, graph paper was an option but for this intervention it was mandatory as part of augmenting or refining the GD. The questions were changed in that it addressed the edges specifically and not demanded a general method to address the edges. The questions were broken down so that the learner could understand that each side or edge had to be checked individually and that the code according to the x-y coordinates differed. A specific challenge or problem was highlighted,

namely that the learners had to write code to detect if the Camel object touched any side of the World. Code was needed for the specific edges, such as the left, right, top or bottom. The learners also had to describe what they understood by the IF statement.

(b) Analysis and Discussion of Intervention

The learners used graph paper as a given to represent the world on scale. Learners had to mark the coordinates at all four corners of the graph paper. As part of the class discussion, the researcher showed the learners how to enact the graph paper as an electronic exercise, by positioning the object at each corner and inspect the object (Figure 5.49). The `act()` method could also be used as a stepwise debugging option. They understood that the object is a specific size, which had to be considered when verifying the location of the object when turning. Learners performed much better when inspecting the actions through that shown in Figure 5.49. Learners could explain the algorithms with better detail once the steps were enacted. They also represented the object at the correct location on the graph paper because their understanding overall was improved. The very fact of having Moodle available gave most learners an advantage over those that did not use Moodle. Learners were now ready to represent the code as a unit or object using encapsulation within a method by “sharing” the method.

The genetic decomposition of an IF statement could now be indicated on paper in that learners had to be given the opportunity to visualise the steps within such code without abstracting the code in their mind at first. The next step would be to guide the learners on how to abstract the code into a descriptive method performing some action, i.e., an object with the ability to perform actions on other elements within the scenario. All objects playing a role in the solution had to be accommodated, such as `getX()`, `getY()` or `getWorld.getWidth()` and `getWorld.getHeight()`.

(c) Findings

Finding 14D-1: Learners broke down the problem into steps and enacted them using the `act()` method or physical Actions

Finding 14D-2: Learners used graph paper to position the Actor physically with pencil

Finding 14D-3: Learners used abstracted methods to determine x-y positions of the Actor object

(d) APOS discussion

(i) Action

Learners used graph paper or physically moved the object to a position and then inspected the object, as in Figure 5.49.

(ii) Process

Some learners wrote down the steps to enact from memory.

(iii) Object

The IF statement took on form in that a name was given to the group of statements verifying the edges of the World e.g. `turnAtEdge()`.

(iv) Schema

The Schema was expanded.

5.4.3.32 Intervention 14E: The IF statement as a solution to address problems (Appendix P-5)

(a) Description of Intervention

The goal of the intervention was to create a turnkey Greenfoot application. Turnkey is when the Greenfoot application is opened, it is launched. Questions were asked based on previous outcomes in order to force the learner to think about how the object should behave. They enacted the behaviour of the object using code. The scenario now contained trees and a Car object in pre-formatted positions upon starting the Greenfoot scenario.

(b) Analysis and Discussions of Intervention 14E

The learners were given the challenge of writing or creating a turnkey application. That is, when the application is loaded, the World and Actor classes are positioned at the correct locations. The problem was to position the Car object and trees when compiled. Learners added code into the Race World class by creating objects for Car and Trees. The challenges were to detect when the CAR reached any side of the World and transform that into x and y coordinates and take action so that the CAR would keep moving within the World. The learners were also instructed to follow the commands in sequence. Learners now saw several specific steps to solve the problem. Learners managed to position the trees and the Car at specific locations using the `addObject()` method.

(c) Findings

Finding 14E-1: Learners enacted the algorithm in their minds as thought processes

Finding 14E-2: Learners could transcribe the problem in context of Greenfoot and the World

(d) APOS discussion

(i) Actions

Learners imposed Actions on an existing Object, namely the IF statement

(ii) Process

Learners could create the scenario and added all the necessary methods.

(iii) Object

The `turnAtEdge()` method became an object as one abstracted name assigned to a number of commands which discover the edges of the World

(iv) Schema

The Greenfoot Schema now became extensive because of all the scenarios that were created by the learners.

5.4.3.33 Intervention 15: Testing Greenfoot to be accepted among teachers (Appendix Q)

(a) Description of Intervention (Appendix A-8.1 to Appendix A-8.3)

The goal was to test the validity of the Greenfoot application thus far. Three official workshops with the WCED teachers in IT were held to guide them towards investigate the concepts. These workshops were officially arranged by Oracle in SA and took place in the Western Cape at specific areas. The researcher as instructor used the learning material of Greenfoot produced by Oracle to convey the Greenfoot concepts.

(b) Analysis and Discussions of Intervention 15

It was evident that some teachers struggled with creating solutions to problems in Greenfoot, which could be attributed to compounding the entire course into a two-day course. However, the lessons were received well and most teachers could relate to Greenfoot through their Java programming language classes. Many of them were teaching Java as programming language at the time and could easily construct code within the Greenfoot editor. The Greenfoot IDE was easily assimilated. The compilation process was not unfamiliar to them and debugging was kept to a minimum as opposed to CAT teachers who have not done any Java programming. The `Act` method was also a new concept to teachers in that it was recursively called during a run phase. Teachers had a sequential idea of code execution where code started and stopped. Their concept of creating an algorithm now changed into that of concurrency.

The most interesting of all outcomes was that many teachers doing IT and programming already had pre-set ideas of variables and coding and found it difficult to understand how the `act()` method is called recursively. The final scenario used with these teachers is captured in Appendix V, mixing music with Greenfoot. Seeing that the teachers already had a solid understanding of arrays, the keys and notes were stored in an array. The sounds of the

different keys were used from an existing scenario in the Book Scenario examples on the Greenfoot site.

The teachers experienced the same challenges as the learners. This highlighted the fact that learners were more acceptable to learning, seeing that certain existing knowledge could prevent the acquisition of new knowledge, also known as 'epistemological obstacle' (Brousseau, 1983). Beliefs about the self, beliefs about social content are a difficult domain to penetrate and to change in an instant (Jankvist & Niss, 2018).

The learners were much more open to learning than their teachers were.

(c) Findings

Finding 15-1: The teachers found the course challenging because of the short timeframe of only two days

Finding 15-2: The teachers could see the relationship between Java and Greenfoot code and applied some Java code within Greenfoot

Finding 15-3: Teachers and learners experienced the same challenges

Finding 15-4: Pre-knowledge prevented acquisition of new knowledge

Finding 15-5: Beliefs about mathematics may also become a limiting factor in making progress in mathematics, unless some meta-belief system is implemented

Finding 15-6: Learners were much more open to Greenfoot, possibly because of pre-knowledge and beliefs, which were more visible in adults

(d) APOS discussion

(i) Actions

The approach was that of enacting code at first. The researcher/instructor focused on worked-examples to optimise time as opposed to constructivist learning.

(ii) Process

Teachers who taught programming showed more potential in terms of creating methods that encapsulated code fragments into abstract concepts.

(iii) Object

Programming teachers already had control structures and variables as part of their Object repository. Their adaptation was more focused on assimilation than accommodation of new concepts.

(iv) Schema

Programming teachers had a programming Schema to assimilate where other teachers with a CAT background had to accommodate new concepts. These CAT teachers commented that the course was overwhelming, but doable, owing to the worked-examples.

(e) Summary

Appendix R-4 highlights the interviews conducted with learners on the Mathematics and Science interventions done.

Chapter 5 reported on the EDR data collection process and analysis of the data. The design was based on a mixed methods approach, where interventions and interviews were used to collect data. Qualitative data analysis was the focus of the analysis. The EDR approach gave the researcher a means to investigate the wicked problem at hand, namely the development of computational thinking among grade 8 and 9 learners of a private school. The researcher explored the use of Greenfoot as programming language to develop computational thinking among learners, with APOS theory as lens. Understanding of APOS gave learners clarity of enhancing computational thinking during basic programming in Greenfoot. The findings show that there is a relationship between computational thinking and the application of Greenfoot as programming language. Furthermore, learners do have the ability to apply APOS theory, as used in mathematics learning, using the Greenfoot programming language through specific applications and tasks. The analysis further shows that learners are better equipped to apply computational thinking outside their system about mathematics. It is difficult to change an existing belief system such as applying APOS theory whilst learning mathematics. A programming language such as Greenfoot effectively highlights the phases of APOS theory in a tangible manner when applied in a constructionist manner.

5.4.3.34 Intervention 16: Creating an Arcade Game (Appendix U)

(a) Description of intervention (Appendix U)

The goal was to let learners construct a more complicated arcade game. The learners were briefed in terms of Appendix U. The game was supposed to be done in 2014, but because of the distracting activities at the end of year, the researchers forwarded this to 2015. The learners had to construct an upright rectangle for the Ping-Pong game. They had to construct the paddle and the ball. They also had to move the paddle using the left or right arrow key to move it horizontally. The researcher also produced videos of the game to help the learners construct the game.

(b) Analysis and Discussions of Intervention

Learners participated selectively. The researcher did not force any learners to participate. Only those learners with a more in-depth understanding of Greenfoot thus participated.

These learners saw the step-by-step development and had enough pre-knowledge on adding objects such as the paddle and ball to the Green world. The learners used the Randomize method to directing the ball in any randomized direction above the paddle. The learners only worked up to the ball connecting to the paddle and then changing direction or bouncing off.

(c) Findings

Finding 16-1: The learners were able to use their Schema of Greenfoot to relate

Finding 16-2: The learners managed to bounce back the ball

Finding 16-3: The game was incomplete due to time constraints

(d) APOS discussion

(i) Actions

Learners watched videos on the game to enact every step.

(ii) Process

Learners relied on “met-befores” to construct the basic framework of the game.

(iii) Object

Similar methods such as `turnAtEdge()` was constructed to guide the ball. The object was the Game. This was the structure for the learners when they referred to the construction of a game through coding.

(iv) Schema

Assimilation and accommodation took place among the participant learners.

(e) Summary

The Ping-Pong game showed that programming concepts can be used as “met-befores” to construct a game. Time was however a limiting factor.

5.4.4 Interviews

5.4.4.1 Interviews: Algebraic Simplification and the Electrical Circuit Diagram (Appendix R)

Interviews were done at end of the research period. A summary of the findings and themes of the interviews are depicted in [Table 5.11](#) and the themes are depicted in

[Table 5.12](#). Learners were given the sheets in Appendix R-1, R-2 and R-3 beforehand. These examples were Mathematics and Science assessment examples that were taken down at the time in the private school and public schools. The learners were therefore familiar with the questions and level of work. The learners were to be assessed on this work again and all

interviews occurred before the assessment. Each interview was recorded using a recording device and the learners were informed of this beforehand. The learner and researcher could conduct the interview in a classroom away from other learners. Learners had a choice to participate and the atmosphere was relaxed and quiet. The researcher guided the interview and allowed the learner to select the problem equation for discussion. The researcher made notes where the learner added gestures that could not be recorded.

5.4.4.2 Interview IA: Algebra Exercise on Simplification (Appendix R-1)

(a) Description

Interview IA consists of a selection of mathematical algebraic expressions that needed simplification. Interview IA was taken from common examples done in mathematics at the private school and given to these learners to establish the status quo of the skillset among learners and their position within the mathematics concepts on algebraic expressions and simplification. The researcher marked the answers and conducted open-ended interviews with learners on those problems that learners struggled with.

(b) Analysis and Discussions

Learners completed the exercises so that the researcher could focus on the problem areas. Learners showed anxiety during the exercise, as indicated in the theoretical conceptual framework.

Learners were still at the Action phase of APOS, as they were locked into using a calculator for any calculation. MDL responded: "Yes, I must use a calculator" (Appendix R-4). This led to still performing Actions and due to not memorising their rules on exponents, the learners were just not able to make progress with visualising objects within the problems, let alone attaching that to a Schema. They therefore struggled with abstraction, like substituting an "a" for $3x$ and a "b" for $2y$ in the expression. Some algebraic expressions were also calculated as a Process without physically writing it down in taking an Action first. This was many a time enforced by the teacher although the learner did not complete the Action phase, as the teacher did not want to see steps. The teacher in this case knew they should not perform at the Action phase anymore intuitively, but that they probably did not understand the APOS theory. The learners never unpacked the algebraic expression into steps and became confused with using the expression as a Process, i.e. memorising multiplication and abstraction of the simplification of the expression, which resulted in errors.

Researcher: What do you do about your problems?

MA: I knew my rules but just forgot them. I am unsure about them. I take extra Master Maths classes. I also have problems with geometry. I simply do not know my rules (Appendix R-4).

Researcher: What is area of rectangle and triangle and so on?

MDL: No I do not know the formulae. I will have to go home and memorise them. Hhm, (quiet) (Appendix R-4).

The learner also calculated squares and multiplication without considering the sign of the number. The important aspect is taking an Action by writing the values down and not just calculating values all at once as a Process. This can only be achieved once the learner understands the steps as part of the Action phase. Even where an expression consists of many x 's and y 's, the learner could not use abstraction by assigning or substituting an alphabetic letter to a sequence of numbers. Learners could not generate any formative questions to explore deeper mathematical concepts, embedded in these basic expressions to simplify.

(c) Findings

Finding IA-1: Most learners could not apply abstraction

Finding IA-2: The majority of answers were incorrect

Finding IA-3: Learners tried to place the Process phase before the Action phase

Finding IA-4: Learners memorised formulae of objects but in isolation and not as an object that has a relationship with another object to form part of a Schema based on a mathematical concept, which points to simplification with indices in this case

Finding IA-5: Learners also worried about the formula rather than the concept definition

Finding IA-6: Most learners were still at the Action phase in APOS, but these actions were also done in silos

Finding IA-7: Learners did not reflect on the problem within the context of algebra or geometry

Finding IA-8: Learners lacked Schemas they could access to solve a problem

(d) APOS discussion

(i) Actions

Learners used calculators for very basic calculations, such as 3×2 .

(ii) Process

Due to calculator usage for basic calculations these timetable activities were never memorised, let alone more difficult processing of exponents or factorisation. No progress to Process phase.

(iii) Object

The “exponent” mathematical Object within algebra was not fixed in the learners’ minds or did not form part of the learners’ skillset or mathematical concept definition on exponents and its rules. What teachers think should be a Process, learners first did not relate to some Action phase.

(iv) Schema

Although a minority of learners possessed some exponent Schema, to name one, the overall Schema that should house these mathematical concept definitions was absent.

(e) Summary

Learners skipped the action phase because the teacher’s expectation was that the learners should not follow action-based approach. The APOS stage of action was forced to become a process and it hampered learning. APOS stages were unknown at private school.

Learners relied on their belief system about mathematics, and the Pop-Ed culture (Papert, 2005) made them cling to the concept image instead of the definition. The silos of concept images created learning challenges because of their beliefs about mathematics. Learners showed that they could link APOS theory in programming and apply that to mathematics but could not find the solutions to mathematics problems because of a lack of abstraction in mathematics.

5.4.4.3 Interview 1B: Electrical Circuit (Appendix R-2 and R-3)

(a) Description of Interview 1B – Electrical Circuit

The learners were given the electrical circuit diagram (Appendix R-2) to infer the answers of questions on this circuit diagram. The exercise concerned batteries or energy cells in series and light bulbs as resistors in parallel. Questions were given on this circuit diagram such as calculating resistance (R) or electrical current (I). The questions were rephrased, e.g., “What is the reading on the ammeter?” or “What is the total resistance in parallel?”

(b) Analysis and Discussions of Task 1B

Learners were prematurely given the Voltage-Resistance-Current pyramid to memorise. When they were confronted by this exercise, the learners made a drawing of the pyramid on their paper. Learners were taught at all three schools, within the ambit of the research, to draw a triangle with VIR as depicted in Appendix R-3. This provided a concept image as a practical plan initiated by many schools to focus on better marks as opposed to better understanding. Unfortunately, this did not enable learners to grasp the scientific and mathematical concept as per definition for electrical circuits. Although learners performed

rote learning in total isolation, they could answer most questions using the pyramid and a calculator.

The learners easily inferred the answers, but no real understanding of the real-world problem emerged from questions posed to these learners afterwards. The learners thus showed no connection with the circuit diagram and its properties other than applying the pyramid. The relationship between learner and circuit diagram was absent. The individual terms like $V =$ Volts and so on, did not appeal to most of these learners.

(c) Findings

Finding IB-1: Learners followed a rote learning technique

Finding IB-2: Learners calculated the correct answers

Finding IB-3: Learners lacked a connection to the subject and electrical circuit diagram topic

Finding IB-4: Learners were lost without the pyramid

(d) APOS discussion

APOS and learning were the main themes.

(i) Actions

Most learners enacted the task by using the pyramid as an aid. Let this be a “calculator”, as Actions are taken with basic calculations using the formula. Even the pyramid suggested what must be divided and what should be multiplied, having the V at the apex and I and R at the base.

(ii) Process

Due to pyramid usage for basic calculations, these fractional parts, such as $V=IR$, could not be deduced/inferred into $I = V/R$ and so on without the pyramid available to learners. The actual manipulation of V , I and R did not realised as a science concept definition and the learners also never memorised the Actions.

(iii) Object

The “electrical circuit” as a Scientific Object was absent, together with the mathematical calculations. Learners were ignorant about discussions on the electrical circuit.

(iv) Schema

Not sure if any learners owned the Schema on electrical circuit diagrams, as the pyramid did not allow any growth or expansion of knowledge.

(e) Summary

Having performed these basic tasks with learners, they were relevant, consistent, effective and practical within the context of computational thinking skills – relevant in that the tasks addressed everyday life concerns of learners; consistent in being a valid construct to perform as a measure to detect computational thinking; effective in that these tasks produced outcomes which motivated further interventions to be executed. Practical for these tasks could be identified and performed by teachers every day in the classroom. It was evident that reflective abstraction forming part of computational thinking within mathematics was still a challenge to most learners. It is therefore a concern that these everyday tasks place most learners at an Action level only. Learners still have progress to be made in order to achieve Schema status on mathematical concept definitions. Revisiting Appendix R-4, interviews were held on the problems the learners struggled with initially. Greenfoot learners could immediately position themselves within the APOS theory of mental structures and thought about the mathematical problems differently. Differently points to opting for different ways in solving the mathematical problem as opposed to just providing a wrong once-off answer.

A few discussion classes were held with 8 learners to address mathematics. From the questionnaires they answered, the learners displayed a lack of insight into solutions they would have thought about in terms of Actions that they could perform to solve the questions. The fact that a learner stated she did not recall the difference between two squares, shows that no Process was formed after the teacher’s explanation of the mathematical concept. This created a blockage with discovering or building the Object for this mathematical concept and no Schema could be created as a long-term goal.

Table 5.11: Summary of the findings and themes of the interviews

Findings	Theme
Learners skipped the Action phase in mathematics because teachers told them to	APOS/LE
Learners preferred concept images in mathematics above concept definitions	LE/BE/CB
Learners followed recipes that provided answers without understanding	Abstraction/ Computational Thinking
Learners could not cope without a recipe	LE

Table 5.12: Themes from Interviews

Themes	Abbreviation
APOS	APOS
Learning	LE
Beliefs	BE
Cognitive Balance	CB

5.4.5 Phase 3: Assessment

5.4.5.1 Themes within interventions

The 98 findings derived from the 34 interventions were summarised,. Categories were created and reduced to 8 themes, depicted in [Table 5.13](#). Further assessments together with the interviews in section [5.4.4](#) are discussed in section [6.2](#).

Table 5.13: Grouping of findings, summary of findings, categories and themes

Find#	Summary Findings	Category	Themes	
14D-2	Learners went back to action to understand problem	Action	APOS	
14B-3	Learners discovered how to debug step-by-step	Action		
14A-2	Learners went back to action to understand problem	Action		
14B-1	Learners re-enforced understanding through action	Actions		
16-1	Learners illustrated APOS qualities	APOS		
3B-3	Learners illustrated APOS qualities	Process		
5B-3	Learners' schema was better developed	Schema		
8-3	Learners' schema was better defined in their minds	Schema		
2-4	Learners' schema for maths had to be re-thought	Schema		
13-1	Learners enhance understanding in Visual programming language	Schema		
13-2	Learners' schema became important	Schema		
14A-3	Learners' schema became important	Schema		
14A-4	Learners' schema became important	Schema		
14C-1	Learners had urge to move through Schema stages	Schema		
14E-1	Learners' Schemas played prominent role in their understanding	Schema		
14D-2	Learners' schema expanded	Schema		Beliefs
14B-3	Learners' schema expanded	Thoughts		
15-5	Beliefs created challenges	Beliefs		
15-6	Beliefs help with learning	Beliefs		
15-4	Teachers made linkages with "met-befores"	Met-befores		
3C-6	Negativity towards programming language due to unknown as in maths case	Met-befores		
2A-2	Learners fixate on concept images	Pop-Ed	Cognitive Balance	
2B-3	Learners are not keen to to explore	Cognitive Load		
2C-1	Learners welcome LMS as resource	Cognitive Load		
2C-2	Learners welcome LMS as resource	Cognitive Load		
3A-2	Learners had too much to memorise	Cognitive Load		
3B-1	Interest in a topic generates positive attitudes	Cognitive Load		
3B-2	Interest in a topic generates positive attitudes	Cognitive Load		
2-1	Learners are not keen to to explore	Cognitive Load		
2-2	Learners are not keen to to explore	Cognitive Load		
14C-2	Learners used coding to describe algorithm	Cognitive Load		
14E-2	Learners used coding to describe algorithm	Cognitive Load		
2A-1	Learners want to explore work they understand	Met-befores		
1-1	Learners used enactment to avoid abstraction	Abstraction		Computational Thinking
1-2	Abstraction lacks from learners performing mathematics	Abstraction		
3C-4	Learners show lack of knowledge and skills	Abstraction		
1-3	Learners show lack of knowledge and skills	Abstraction		
6-1	Teachers found topic challenging	Abstraction		
9-1	Learners applied abstraction through encapsulation	Abstraction		
9-2	Learners enhance understanding in Visual programming language	Abstraction		

Find#	Summary Findings	Category	Themes
10-1	Learners used built-in method to solve problem	Abstraction	
10-4	Learners used built-in method to solve problem	Abstraction	
11B-1	Learner linked programming language and Windows Tools	Abstraction	
11B-2	Learners applied abstraction through encapsulation	Abstraction	
14D-3	Learners used built-in method to solve problem	Abstraction	
6-2	Learners followed APOS	Process	
8-1	Learners followed APOS	Process	
3C-1	Learner links absent between mathematics in Greenfoot	Relation	
8-4	Learner linked mathematics in Greenfoot	Relationships	
14A-1	Learners revisited GD	Actions	Learning
15-3	Teachers had similar challenges than learners	Met-befores	
15-2	Teachers made linkages with "met-befores"	Relationships	
3B-4	Learners fixated on concept images	Skill	
5A-2	Technical challenges influenced learning	Teaching	
5A-3	Technical challenges influenced learning	Teaching	
5A-4	Technical challenges influenced learning	Teaching	
11A-1	Learners' academic world must be structured	Teaching	
12-4	Learners' academic world must be structured and official.	Teaching	
15-1	Teachers found topic challenging	Teaching and Learning	
3A-3	Learners could work on their own if they were given guidelines	Teaching and Learning	LMS
3C-5	Learners show lack of knowledge and skills	Teaching and Learning	
2B-1	Learners are not keen to to explore	Moodle	
2B-2	Learners are not keen to to explore	Moodle	
3A-1	Learners liked a change in behaviour	Moodle	
4B-1	Moodle solved challenges	Moodle	
4B-2	Moodle solved challenges	Moodle	
4B-3	Moodle has costs	Moodle	
4B-4	Moodle solved challenges	Moodle	
5B-1	Moodle solved challenges	Moodle	
5B-2	Moodle solved challenges	Moodle	
11A-2	Learners used Moodle for preparation	Moodle	
3B-5	Moodle solved challenges	Moodle	
16-2	Learners' coding enhanced	Coding	Programming Language
16-3	Learners' coding time intensive	Coding	
3C-2	Learners lack programming language knowledge	Coding	
3C-3	Learners lack programming language knowledge	Coding	
5A-1	Learners enhance understanding in Visual programming language	Coding	
7-1	Learners enhance understanding in Visual programming language	Coding	
7-2	Learners enhance understanding in Visual programming language	Coding	
8-2	Learners enhance understanding in Visual programming language	Coding	
8-5	Learners enhance understanding in Visual programming language	Coding	
9-3	Learners see value of control structures	Coding	
9-4	Learners had challenges to understand execution of Greenfoot	Coding	
9-5	Learners enhance understanding in Visual programming language	Coding	
2-3	Leaners show challenges with IDE of programming language	Coding	
10-2	Learners enhance understanding in Visual programming language	Coding	
10-3	Learners enhance understanding in Visual programming language	Coding	

Find#	Summary Findings	Category	Themes
11B-3	Learners enhance understanding in programming language	Coding	
12-1	Learners enhance understanding in programming language	Coding	
12-2	Learners enhance understanding in programming language	Coding	
12-3	Learners enhance understanding in programming language	Coding	
12-5	Learners found syntax challenging in coding	Coding	
13-3	Learners enhance understanding in programming language	Coding	
14B-2	Learners enhance understanding in programming language	Coding	
4A-3	Technical networking allows learner external access	Networking	TC
4A-1	Technical wizardry can save costs	Technical	
4A-2	Technical logic can secure productivity	Technical	
4A-4	Power failures highjack technical expertise	Technical	

Table 5.14 links the findings with the problem statement, research questions and research objectives.

Table 5.14: Relationship of research questions, objectives, findings, main findings and themes

Problem Statement	RQs	Objective	Themes [Findings]	Main Findings
It is unclear how computational thinking can be promoted among high school learners at a cognitive level of formal operations	RQ 1: What are the characteristics of an enhanced learner's teaching and learning strategy that can empower learners to master computational thinking skills through APOS theory, infused by a programming language at high school level?	To explore and understand the characteristics of a programming language which promote computational thinking through APOS theory, at a cognitive level of formal operations	Computational Thinking, Programming Language, TC, APOS	1-1, 11B-2
	SRQ 1.1: What factors are needed to promote computational thinking at a cognitive level of formal operations among high school learners?	To determine the factors which inform computational thinking among high school learners at a cognitive level of formal operations (CLFO)	Computational Thinking [1-1, 1-2, 1-3, 3C-4, 9-1, 11B-2], Programming Language [9-2, 10-1, 10-4, 14D-3], TC [11B-1]	Computational Thinking & Programming Language 1-1, 9-2, 11B-2
	SRQ 1.2: What type of programming language may be used to promote computational thinking skills at a cognitive level of formal operations?	To determine the characteristics of a typical programming language that may promote the cognitive level of formal operations (CLFO)	Programming Language [3C-2, 3C-3, 2-3, 5A-1, 7-1, 7-2, 8-2, 8-5, 9-2, 9-5] LE [14E-2]	5A-1, 3C-2
	SRQ 1.3: What constructs within the programming language facilitate APOS theory at a cognitive level of formal operations?	To determine commonalities of constructs in APOS and the programming language	APOS [17-1, 3B-3, 14A-3, 5B-3], LE [2-4, 13-1, 13-2, 14C-3, 14C-1, 14D-2, 14E-1] Programming Language [8-3, 14A-2, 14B-3, 14D-1]	
	RQ 2: How can computational thinking skills at a cognitive level of formal operations be promoted among high school learners through the teaching of a programming language aligned to Action Process Object Schema (APOS)?	To explore and understand how a programming language aligned with APOS theory promote computational thinking at the cognitive level of formal operations (CLFO) for high school learners	Programming Language, CB, LMS, LE	

Problem Statement	RQs	Objective	Themes [Findings]	Main Findings
	SRQ 2.1: How are the constructs of a programming language taught among high school learners at a cognitive level of formal operations?	To explore and understand how constructs of a programming language facilitate high school learners at a CLFO	Programming Language [2-3, 3C-2, 3C-3, 5A-1, 7-1, 7-2, 8-2, 14B-2]	
	SRQ 2.2: How do the constructs of a programming language align to APOS among high school learners at a cognitive level of formal operations?	To determine higher-level constructs within a programming language which promote APOS among high school learners	Programming Language [2-3, 8-5, 9-2, 9-3, 9-4, 9-5, 10-1, 10-2, 10-3, 11B-3, 12-1, 12-2, 12-3, 12-5, 13-3, 14B-2, 14D-1, 14D-3]	
	SRQ 2.3: How does the use of an LMS, as a platform for learning, aid the teaching of a programming language aligned to APOS to promote computational thinking skills at a cognitive level of formal operations among high school learners?	To combine the usage of an LMS and a programming language in order to assist high school learners with “worked examples” of advanced higher-level constructs in a programming language and cognitive load theory (CLT)	CB [2A-1, 2C-1, 2C-2, 14C-2] LMS [2B-1, 2B-2, 3A-1, 3B-5, 4B-1, 4B-2, 4B-3, 5B-1, 5B-2, LE [11A-2]	2A-1, 14C-2 CB

5.4.5.2 Themes within interviews

The twelve findings of the interviews were summarised, as indicated in [Table 5.15](#). Categories were created and reduced to five themes, depicted in [Table 5.15](#). The themes are discussed in Chapter 6, section [6.2](#).

Table 5.15: Grouping of interviews, summary of findings and themes

Find#	Summary Findings	Themes
IA-1	Most learners could not apply abstraction	APOS
IA-2	The majority of answers were incorrect	BELIEFS
IA-3	Learners tried to place the Process phase before the Action phase	APOS
IA-4	Learners memorised formulae of objects but in isolation and not as an object that has a relationship with another object to form part of a Schema based on a mathematical concept, which points to simplification with indices in this case	LEARNING
IA-5	Learners also worried about the formula rather than the concept definition	BELIEFS
IA-6	Most learners were still at the Action phase in APOS, but these actions were also done in silos	COMPUTATIONAL THINKING
IA-7	Learners did not reflect on the problem within the context of algebra or geometry	LEARNING
IA-8	Learners lacked Schemas they could access to solve a problem	APOS
IB-1	Learners followed a rote learning technique	LEARNING
IB-2	Learners calculated the correct answers	APOS
IB-3	Learners lacked a connection to the subject and electrical circuit diagram topic	COGNITIVE BALANCE
IB-4	Learners were lost without the pyramid	LEARNING

5.5 Summary

This chapter covered the data collection process and analysis of the data. A qualitative design was used through abduction on data obtained from interventions within the EDR. The

data were interpreted using a qualitative analysis based on the FEDS framework and reflected on using the Gregor, Müller and Seidel (2013) framework. The FEDS framework (Venable, Pries-Heje & Baskerville, 2016) (Appendix U) highlights two approaches, formative or summative evaluation, on account of any of four suggested evaluation strategies. The strategy used four steps, namely: (i) explicate the goals of the evaluation; (ii) choice of the strategy; (iii) choose the properties to evaluate; and (iv) design the evaluation phases.

The data analysis and discussion are done in Chapter 6 by applying the Gregor, Müller and Seidel (2013) framework.

CHAPTER 6: DISCUSSION

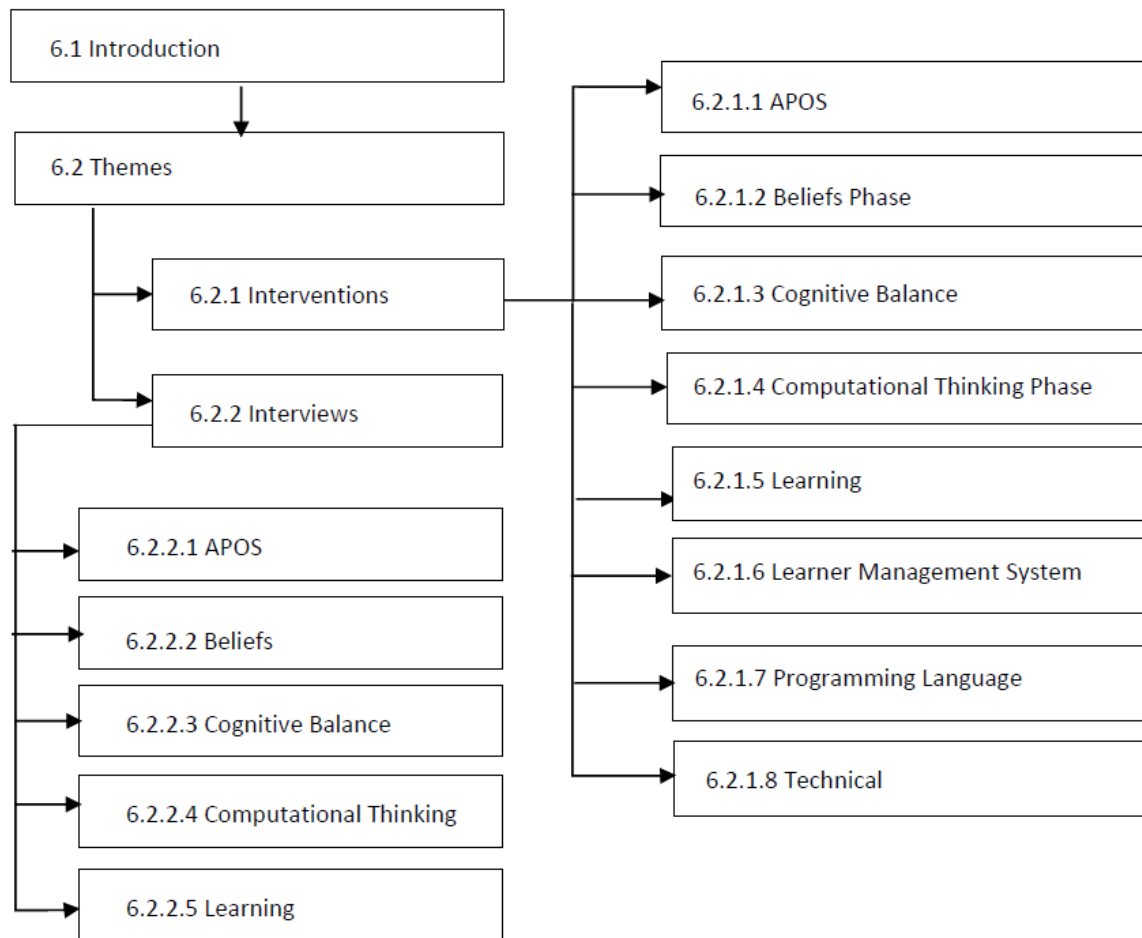


Figure 6.1: Chapter 6 Layout

6.1 Introduction

The low pass rates of learners enrolled for Mathematics and Science at high school level in SA are of great concern (Reddy et al., 2015; Voogt et al., 2015; Reddy, 2014; Spaul, 2013) (see section 1.1 for more information). This research proposes a conceptual framework (section 8.4, Figure 8.3) that can be used to improve computational thinking of learners using a programming language and LMS in applying APOS theory. Computational thinking is embedded in reflective abstraction as found in APOS (Cetin & Dubinsky, 2017). Research on APOS theory worldwide shows its positive contributions to the success rates of mathematics of learners (Arnon et al., 2014) and is therefore used as a theoretical lens. Chapter 6 (Figure 6.1) is presented as (i) themes from the interventions and (ii) themes from the interviews and observations.

6.2 Themes

6.2.1 Interventions

A total of 8 themes were constructed after subdividing the findings into categories (Table 6.1). The themes are APOS, Beliefs, Cognitive Balance, Computational Thinking, LMS, Learning, Programming Language/ Coding, and Technical Themes (Appendix Y).

Table 6.1: Summary of findings per theme

Theme	Number of Findings
APOS	17
Beliefs	5
Cognitive Balance	11
Computational Thinking	16
Learning	12
LMS	11
Programming Language/Coding	22
Technical	4

These themes will now be discussed according to Table 6.1.

6.2.1.1 APOS theme

The APOS theme comprises of 17 findings (Table 5.13). The APOS theory supports mathematical problem solving and computational thinking through reflective abstraction (Aho, 2012; Selby & Woollard, 2014; Cetin & Dubinsky, 2017; Denning, 2017). These mental structures are enforced among learners to acquire a sense of the phases.

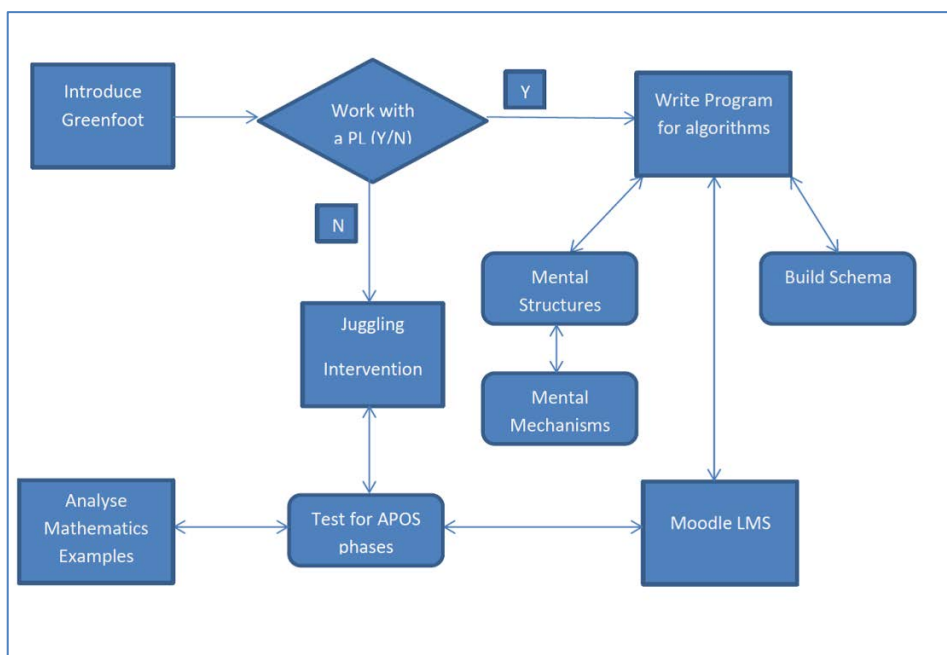


Figure 6.2: Flow of APOS theme

Within these phases, the mental mechanisms (section 2.2.2.1(c)(vii)) were applied for each mental structure. Figure 6.2 depicts the process that was applied to introduce the Greenfoot programming language and APOS theory to learners. Learner non-compliance forced the selection option as indicated in Figure 6.2, which determined whether learners worked with a programming language before or not. Learners found the Greenfoot programming language challenging, as indicated in Intervention 2 (Appendix C), through formative evaluations (Gregor, Müller & Seidel, 2013) using observation. Learners were unable to step directly into the Greenfoot programming language using the IDE, which necessitates literacy in the Greenfoot programming language. The researcher had to rethink the approach, considering the findings of Intervention 2. The theoretical conceptual framework needed a revision, and a GD was necessary for teachers and other researchers, which demanded enhancements, as rolled out to the learners through activities, classroom discussion and exercises (ACE, Figure 2.20).

Updating the theoretical conceptual framework transpired through ACE. Intervention 3, namely the “Juggling” intervention, illustrates the practicality of APOS theory to learners when they do not yet have the cognitive structures to use Greenfoot programming language to write programs. Four formative evaluations were enforced by the Greenfoot programming language introduction.

The APOS theory mental structures which make up the APOS theme will now be discussed. It must be kept in mind that the APOS theory uses mental mechanisms (Figure 2.17), where an Object can be broken down into steps using reversal to return to the Action phase or where interiorisation is applied by learners when an Action phase goes forward into a Process phase to build a Schema through generalisation (Dubinsky, 1991; Arnon et al., 2014).

(a) Action Phase

The Process phase in APOS theory was absent for Intervention 2, which necessitated the Action phase to be the initial step in the Greenfoot programming language rollout using a GD informed by the fourth formative evaluation (Figure 2.20) and depicted in Table 2.1. The reversal mental mechanism is used by introducing the “Juggling” intervention 3. Intervention 3 informs APOS theory in a practical manner, which evolved from the basic steps during the Action phase into “Juggling” as an Object that builds onto the Schema, where learners generalise the processes to juggle into “Juggling” as an object. The “Juggling” object has different interpretations for different learners and is depended on their involvement to juggle, as supported by the questionnaire handed out to learners during the intervention. These different interpretations led to learner challenges during Intervention 3, which were affected by the extent to which learners fulfilled the “Juggling” activity through ACE. ACE was applied

through the intervention, the Moodle step-by-step guidance in the form of a video, and the exercise performed in class, on the playground and at home. The juggling intervention created an awareness of APOS theory through practice, which gave learners a “met-before” to approach the Greenfoot programming language with APOS as lens. Moodle as LMS and the GD (Intervention 2) gave learners an understanding of how “juggling” can be perfected. The success of the intervention became the learners’ choice. The findings of Intervention 2 show that learners did not follow instructions to complete homework. Furthermore “met-befores” guides learners in their search for knowledge. The GD process (Appendix D-1) acts as a framework for teachers and learners. The GD on “Load a Greenfoot Scenario” (Appendix D-2) plays a pivotal role in the tasks of the learners, because it gives clarity on how to approach the “Juggling” task. The interventions are of high quality, as relevance, consistency, practicality, and effectiveness are guaranteed when looking at the outcomes produced. The progression path is described by making learners aware of the phases of APOS and emphasising that coding can be achieved through the GD within a programming language. The teachers are also given a specific guideline in Appendix D-1 on how to structure a GD and its rollout through the ACE (Intervention 2B, [Figure 2.20](#)) process as discussed above.

The juggling exercise instilled thought processes in learners, which is found in computational thinking, according to Selby and Woollard (2014), such as memorising the steps, and not having to physically think about the tossing of a ball. The involuntary action only became a reality through practice. Learners realised that they could not think about the action every time the ball is tossed. Neither could they complete the juggling without watching the video on Moodle. Learners stated in the questionnaires (Appendix E-2, Intervention 3B), that their performance would have been better if they watched the video prior to the exercise, and points to a personal choice. As indicated in [Figure 6.2](#), the researcher ventured back into current mathematical problems that learners were confronted with to link that with APOS through learner interaction by means of a questionnaire (Appendix E-3).

Mathematics (Appendix G-2, Intervention 5B) was introduced with a discussion on the APOS impact versus that of “juggling”. Learners were viewing mathematics problems through an APOS lens. The learners saw memorisation of actions needed to revisit steps to complete certain actions, which apply to simplification or juggling. This is also called the Action phase of APOS, where learners needed to identify and show the steps towards a problem.

(b) Process Phase

When these steps are memorised, learners approach the Process phase. Learners were given simple tasks such as login credentials (Intervention 4B) that needed to be memorised. All these Actions reverted back to the APOS theory, and so they acquired a better

understanding of the APOS acronym. APOS theory started to become a frame of reference. Intervention 8 focuses on the Greenfoot programming language code through the APOS theory as lens. The Moodle LMS provided significant improvement among learners because they could acquire the skills to create a scenario through flipped classroom techniques as shown in Moodle. The researcher observed that time was shortened in creating scenarios in the Greenfoot programming language during sessions.

The learners approached a problem (Figure 5.37) and had to derive an algorithm for the problem in populating the scenario as a chess board. The problem created thought processes which strengthened the Process phase in APOS. Thought processes involved mathematical calculations assessed in the outcome of the chessboard in the Greenfoot programming language. These algorithms then formed part of solutions using Greenfoot methods. Learners applied APOS theory techniques in drilling down into complex issues by applying Actions to the complex issues to understand the issues before abstracting. The exercise involved using variables and understanding x and y values used when a balloon moved horizontally or vertically. Some learners tried to venture into a process without taking action first, with dire consequences. Actions entailed learners using the built-in features of Greenfoot to physically establish the coordinates of the object on a chessboard (Figure 5.36) by moving the turtle object on the board and show its x and y coordinates. In doing this, the learners became familiar with the direction of movement, influencing the x and y values to increase or decrease. Having taken these actions, learners can now translate these actions into code, by memorising the outcomes of these movements and the commands that cause the movement into the `act()` method. Figure 5.39 illustrates the Process phase of the learner when he/she writes the code to manipulate the actor in question. To further strengthen the Process phase, learners were also given graph paper to show their understanding of the problem in question, which was to move the Actor object in the World, bouncing off the edges of the World (Figure 5.41). The mental mechanisms emerged as learners plotted the Actor objects on graph paper that resembled the Greenfoot programming language screen or world; once again the turtle-actor-object-movement is reversed, by taking action using graph paper and reconfigure the scenario itself.

After the December holiday break, the learners restarted the Greenfoot programming language classes. The intervention revisited the steps in building a scenario and added code to strengthen the existing Schema that the learners developed in Greenfoot. The Process phase in APOS theory is well executed, which shows a movement in the thought processes of learners. The mental mechanism used by learners is called interiorisation. Learners are now less dependent on taking action first, that is, being at the Action phase; they perform tasks in the mind instead of following physical steps. The Process phase of the learners was

supported by the answers on the questionnaires handed out to the learners. They had to describe the exact execution that they had to follow to get the scenario going. Their answers show insight into Processes. The researcher established, through observation during coding sessions in class and from the answers on the questionnaires completed by the learners, that Actions were interiorised as a Process, as the learners easily described these actions on paper as well, i.e., projecting their thoughts on paper. The code learners wrote on paper show that they could break down the process into steps, also known as reversal (mental mechanism) of the interiorised steps. This was also supported in the decline of watching videos as a source to discover techniques for building a scenario.

(c) Object Phase

Having the Action and Process phases embedded within the learners' thoughts, the Object and Schema phases are of essence.

Control structures such as the IF statement was illustrated by the researcher through interventions 9, 10 and 11. Learners could abstract at a higher level such as creating methods to nest common code including the control structures. When looking at [Figure 5.42](#) where code was produced by learners to allow the turtle Actor object detecting the left, right, upper and lower boundaries through x and y values, using built-in functions, learners embedded that code into a method called `atWorldEdge()`, as depicted in [Figure 5.42](#). When the researcher referred to `atWorldEdge()`, learners immediately, through the reversal mental mechanism, broke down the encapsulated method of code into code steps. Learners used their own descriptions for such methods they produced through a mental mechanism called encapsulation. This encapsulated method of code can be classified as an Object or small piece of the bigger Schema. The Object phase was supported by videos on Moodle and learners could spend as much time out of class on Moodle to perfect their understanding, as it is their choice.

(d) Schema Phase

The researcher, through intervention 11B, established that more learning took place than anticipated. The assessment entailed a project where learners worked in groups of two and practiced constructionism at its best. Other aspects such as MS Paint used by learners to create a World and Actors show that the thought processes of learners expanded into research. Learners also created sound to align with the project. Computational thinking consists of thought processes, abstraction and decomposition (Selby & Woollard, 2014). The thought processes were apparent through videos the researcher took and observation of learner enthusiasm during the project. Abstraction was shown in the outcome of the project as a working scenario that addressed the problem within the assessment. The underlying code produced the abstracted view of the problem. Decomposition was applied in coding

throughout. Even Wing's (2006, 2008) definition of computational thinking as the mental and metal tools in abstraction and automation, became reality in that the problem was abstracted using coding and the code executed (automation) on a platform using the Greenfoot programming language as computational notation (Aho, 2012; Denning, 2017). This was achieved through computational steps that were constructed through thought processes using a computational notation and not by following a sequence of steps (Denning, 2017). The learners' Schema for using Greenfoot programming language to solve problems was sufficient to expand their Schema according to the assessment outcomes satisfying the APOS theory and theme.

6.2.1.2 Beliefs theme

The Beliefs theme comprises five findings (Appendix Y). Learners have a tendency not to explore, and they find it easier to memorise concept images without thinking about the concept definitions, also supported by Arnon et al. (2014:13). This tendency is also apparent in the annual mathematics results (Reddy et al., 2015; Voogt et al., 2015; Spaul, 2013; CDE, 2014). This research shows that the Pop-Ed culture (Papert, 2005) is rife among learners in that they are insecure in terms of the solutions they produce. The memorising can be because of the cognitive load that influences what learners can absorb during class hours and can create a positive attitude among learners (Mostyn, 2012; Papavlasopoulou, Giannkos & Jaccheri, 2019). The impact of "met-befores", as discussed in section 2.3.1 and section 5.4.3.3, Finding 2-2, can also play a role in this phenomenon. According to the literature study (Moscucci, 2007; Bormanaki & Khoshhal, 2017), and through the results of the practical interventions in this research, beliefs about mathematics dominate the learner's mind. The beliefs of learners regarding mathematics influence other beliefs such as programming when the domain, such as mathematics, informs ill theories about programming in this instance. Answers of learners in questionnaires made a clear statement about the state of mathematics and that a programming language is 'not for them'. Although learners also stated that they never used a programming language, they have already formed a negative opinion of a programming language. In reflecting on these opinions, beliefs are highly likely to create negative feelings towards subjects or related knowledge domains and these feelings are sometimes the result of mirror neuron activity, as supported by Moscucci and Bibbo (2015) in section 5.2. These negative attitudes which learners stated in the questionnaires point to unconscious and invisible components that create pre-set ideas among learners about a programming language and mathematics. However, the programming language being unknown to learners, compared to mathematics being known, the programming language as a meta-cognitive system (Flavel, 1976; Moscucci, 2007; Jankvist & Niss, 2018) may influence the thoughts on APOS theory to allow for learning to take place negatively or positively. The programming language meta-cognitive system is a

new approach to this research, which should be dealt with carefully and properly, because it can influence learners' thinking in future endeavours and form part of the learners' belief system. This may also be one of the many reasons why learners do not perform in mathematics (Reddy et al., 2012; Spaul, 2013; CDE, 2014; Reddy et al., 2015). Spaul (2013) describes this as the way in which learners interact with mathematics does not deliver positive outcomes. The programming language approach and the usage of EDR to resolve this wicked problem paves the way for life-long learning and can be damaging if it is not properly and carefully executed (Rittel & Webber, 1973; Camillus, 2008; Peters, 2017; Termeer et al., 2019).

6.2.1.3 Cognitive Balance theme

The Cognitive Balance theme comprises 11 findings (Table 5.8). The cognitive load can be a limiting factor for any learner during coding (Mostyn, 2012; Papavlasopoulou, Giannkos & Jaccheri, 2019) and cognitive load theory is a focus of this research by propagating a constructionist approach during EDR. The instructional approach of learners was changed through inclusion of Moodle, thus allowing learners to have a hub to find solutions to problems which balanced the cognitive load. This promotes the learning efficiency in providing resources to maximise research among learners (Mostyn, 2012).

The cognitive load balance became part of the theoretical conceptual framework through the literature study (section 5.4.3.9). Intervention 3C (Appendix E-3) acted as a trigger to investigate the concept of cognitive load theory and was hence included into the theoretical conceptual framework. The questionnaires provided insight into the instructional resources that needed adjustment, as the findings show that no structure was put in place towards the acquisition of computational thinking among learners. Learners need to organise their working-memory properly. Too many distractions are available to learners through cell phones, noise and any other factors that can lead to a loss in concentration, which may distract their attention. This is called the extrinsic load, also known as the extraneous load (Mostyn, 2012). The learner's working memory should include cognitive load. Learner working memory consists of short- and long-term memory. The cognitive load can reach a balance by keeping the programming language rollout simple, not exposing the learners to too much work and using sensible "met-before" to introduce new concepts (section 2.3.1; section 5.4.3.9). The literature study shows that certain existing knowledge may prevent acquisition of new knowledge (Brousseau, 1983) and that beliefs about the self and social content are difficult to penetrate and change (Jankvist & Niss, 2018). Intervention 15 shows truth in these authors' statements, when Greenfoot programming language was rolled out to teachers. Teachers who were practicing IT and programming had pre-set ideas about programming language concepts and found it difficult to understand the recursive nature of

the `act()` method as an example of the Greenfoot programming language. Learners, on the other hand, were not contaminated with “met-befores” and accepted and understood these concepts easier compared to teachers. When “met-befores” cause conflicts with new contexts, learners may resort to rote learning (Tall, 2004). Rote learning, when using a programming language, is difficult or almost impossible when a learner creates algorithms and produce code for those algorithms. Another structure amidst the constructionist approach to reduce cognitive load, is the understanding and applying of APOS theory. APOS theory was applied during the Greenfoot programming language rollout, as were mathematical concepts in parallel. APOS theory gives learners a structured systems analysis and design approach of any programming language concept that is needed to help create an algorithm to address some problem, which is a positive belief system development in programming. Learners know now how to approach a problem and may ask the right questions to their subject teachers.

However, Schemas are created and stored into long-term memory to allow any learner the use of important concepts as needed. The struggle of learners to abstract and organise learner working-memory seems to be one of the major tasks that needs to be addressed in education. Learners tend to build their trust through memorising concept images which do not form an integrated network of understanding, also known as a Schema. These concepts are floating as silos, which places an extra burden on the cognitive load (Mostyn, 2012; Arnon et al., 2014; Papavlasopoulou, Giannkos & Jaccheri, 2019). This research used essential cognitivist constructs (Mostyn, 2012) to provide instructional resources that support cognitive load theory. Learners seemed to shy away from activities that could promote learning, such as flipped classroom techniques, due to improper memory organisation. Learners still tried to copy their friend’s work and as a result, the work is not embedded in the learners’ long term memory. Learners’ working-memory was enhanced through providing an LMS where information could be stored for the duration of the course. It was observed that the research progressed and learners became more serious about solving problems using this LMS relationship and APOS theory as part of the programming language. Learners relied on Moodle to provide the correct answers as they needed these flipped classroom videos and website references embedded in Moodle to solve problems. Although the LMS was created with content, learners did not always bring the necessary aids to school to accomplish learning, such as headphones as reported by learners in a questionnaire (

[Table 5.7](#), section [5.4.3.9](#)). The responsibility of the individual also played a role in the success of the rollout. The learners were given the assurance that they could do it and with all the necessary help, e.g. the LMS, they understood it was a choice they had to make. Only learners with sound “met-befores” wanted to participate in the exercises. Discipline to follow

instructions was initially lacking in general (Intervention 2: Finding 2-2; Intervention 3A: Finding 3A-2, Appendix E-1). Having rolled out the programming language alongside APOS theory, learners described their code in writing. More complex problems could be enacted in their mind to solve problems, which showed that that higher level of thinking occurred, provided that the cognitive balance was kept optimal through an LMS that aided learners in finding solutions. This phenomenon is also described by Mostyn (2012), namely that the averages of students remain the same during application of cognitive load theory.

6.2.1.4 Computational Thinking Theme

The Computational Thinking theme comprises 16 findings (Table 5.13).

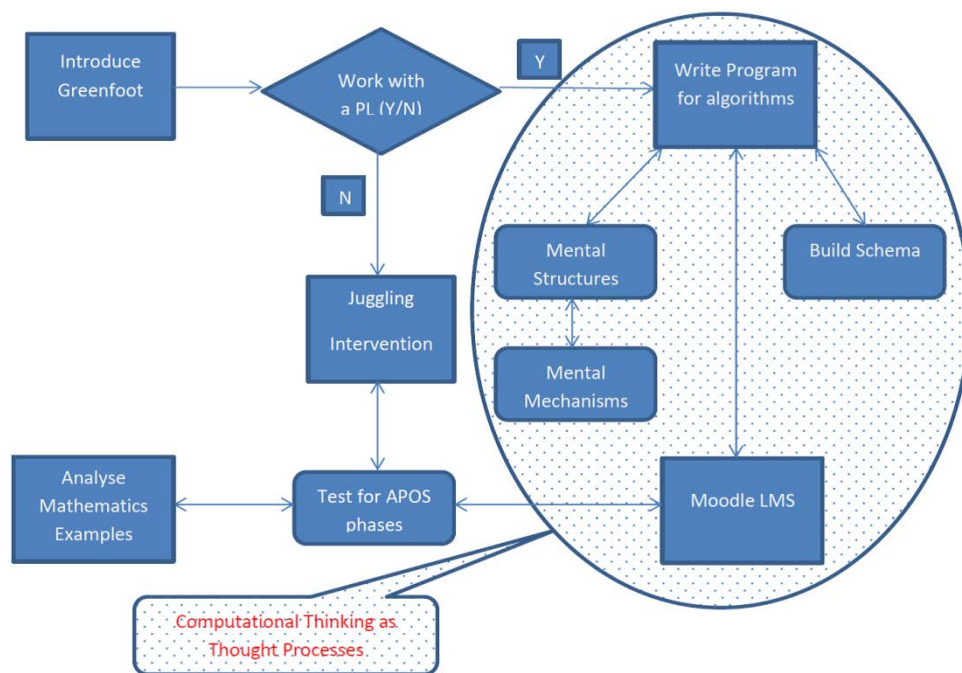


Figure 6.3: Computational thinking hub

Computational thinking consists of abstraction and automation, which, in conjunction with all other discussions on computational thinking, forms a focus point in this research (Figure 6.3). The mental structures or APOS theory supports mathematical problem solving and computational thinking through reflective abstraction (Aho, 2012; Selby & Woollard, 2014; Cetin & Dubinsky, 2017; Denning, 2017).

The interventions and consequent findings show that learners are struggling with abstraction, because they enacted only simplistic actions during these sessions. When the problem is mathematical, learners are incapable of understanding what or how to approach the challenge, as indicated on the questionnaires that were completed by learners. Some tried to punch in numbers on a calculator, which is another Action taken when dealing with mathematics. Others would put forward the answer or jump steps when doing mathematics

as instructed by the researcher and hence the same happens in class when questions are posed to them by their teachers. Questionnaires containing basic mathematics questions were given to learners (section 5.2, Figure 5.3 & Figure 5.4, Appendix B-2), who just gave an incorrect answer, which shows that even though they did not know the answer, they refused to engage in a process of working the question. This shows that learners have not thought about the question, nor have the skillset to calculate the answers. Thought processes thus did not happen, which is a component of computational thinking (Selby & Woollard, 2014). During the research period, the researcher also attended an ongoing workshop between Oracle and Western Cape learners, where most heads of departments in mathematics and computer studies convened for a table discussion. The researcher used the opportunity to ask the teachers whether they were using APOS theory in their mathematics approach; none of them have heard of APOS theory, as indicated through their responses. Being representatives of the WCED, the researcher assumed that teachers have not heard of APOS theory in general. At some stage in the curriculum it is suggested that learners must not implement steps or break down the problem into steps. If learners hardly break down a problem into steps, understanding an abstracted version of a problem remain a challenge. Even simplifying a mathematical equation, as given to learners (Chapter 5, section 5.4.3.17; Intervention 6, Appendix H) may become a challenge to some learners. Although this “ignoring of steps” is advocated by the CAPS documents on mathematics as from grades 7 to 9 by the WCED, the teachers want to enforce this immediately, in accordance with the interviews with learners. The CAPS document is correct, theoretically, and refers to “mental calculations”, but the majority of learners are not ready. Learners want to state the steps of the calculation to ensure a correct answer. Teachers understandably interpret the CAPS document by demanding from their learners to rather perform these “mental calculations” and not write them out. However, APOS theory, on the other hand specifically, contains the Action mental structure, which forms an important component of problem solving through the reversal mental mechanism. This research was not intended to investigate mathematics CAPS documents, but rather to establish an APOS theory approach towards computational thinking.

In the Greenfoot programming language, learners are able to encapsulate code into methods, which is applying mental mechanisms such as encapsulation after interiorising code and approaching problems afresh. Mental mechanisms are abstracted activities and techniques performed by learners, such as embedding code within a method that points to encapsulation (Figure 2.17). These encapsulation techniques are visible in system methods as well as user defined methods. Learners master encapsulating code and then use the abstracted code e.g. `atWorldEdge()`, to accomplish ‘some solution’ as set out in the interventions on coding.

Learners also expressed the need to be informed of assessments, which led to improved preparation and experimentation of the assessment to automate many features in Greenfoot. One such example where learners combined mathematics with coding was when calculating the configuration of a chess board during intervention X in Greenfoot (section 5.4.3.19, Figure 5.36). When learners started out with this research, they stated in a questionnaire that they have challenges in understanding any code, but the manner in which the learners experienced their first encounter laid the foundation of the APOS theory approach. As mentioned during the discussion of the “Beliefs” theme (section 6.2.1.2), much care must go into the developing or nurturing of a programming language, as with the programming language meta-cognitive system in building a programming language-belief, for the programming language belief system of the learner may be damaged in the process, as in the case of the mathematical-belief system. This is also a characteristic of a wicked problem that the process may be more damaging than good, as it paves the way for life-long learning (Rittel & Webber, 1973; Camillus, 2008; Peters, 2017; Termeer et al., 2019).

6.2.1.5 Learning theme

The learning theme comprises 12 findings (Table 5.13). Learning forms an important theme as learners learn in different ways. One of the ways to streamline cognitive balance is to alleviate the tension to find answers to challenges on the internet, also known as the cognitivist theory in psychology (Mostyn, 2012). The Moodle LMS provides a way by which learners could find answers to their problems when coding. This research introduced the term ‘flipped classroom’ (section 5.4.3.15, Intervention 5A) and learners used the option to watch videos and the like. Moodle further allows learners to study whenever and wherever they wanted, which gave them an edge over other learners. Learners showed more confidence to voice their opinion in front of their peers, as noticed in class through observation. They were able to know and find answers outside the classroom, which addresses Papert’s Pop-Ed culture (Papert, 2005). Learners were looking forward to the assessment given to improve their learning, as they investigated and prepared beyond the expected goals for the assessment as done in Intervention 11, Appendix M. The usage of Moodle depends on whether the subject matter was applicable for their course. The GD through ACE acted as a frame of reference when learners got stuck. Learning is influenced by belief systems, which may have a positive or negative outcome in the academic lives of learners, as illustrated with the interventions in this research. The belief system of mathematics for local and international learners has a higher degree of negativity.

6.2.1.6 Learner Management System theme

The LMS theme comprises of 11 findings (Table 5.13). This theme has strong links with the Cognitive Balance theme as well as the Learning theme. Cognitive balance adheres to the

cognitive load theory to minimise a learner's cognitive load (Mostyn, 2012; Papavlasopoulou, Giannkos & Jaccheri, 2019; Sweller et al., 2019) and the LMS supports that. Learning can happen owing to an LMS that satisfies cognitive load theory and provides passage for learners to take. It now became a choice and not a challenge anymore. At the start of this research, learners were confused of where to search (Intervention 2B) and find videos in order to prepare for the classes, especially with issues such as coding (Intervention 2C; Appendix D-1; GD through ACE) and just making sense of the additional burden placed on them. The researcher observed that certain learners initially attended Greenfoot sessions, but without being bothered to learn anything during those sessions. This behaviour may be because learners did not see the research as important towards their schooling performance; they showed characteristics of the Pop-Ed culture (Papert, 2005) and wanted to demonstrate how cool they are; they did not need to do any exercises that would influence them passing the grade. The focus of the researcher's approach had to address learners when they entered the classroom, in providing a source specifically focusing on the Greenfoot programming language and APOS theory.

Learners had to be accommodated within their current "met-befores" or the frustration could have grown among learners. This called for a GD to be created on exactly what was expected of them and how to accomplish the tasks. Appendix D-1 shows the different stages in Schema building and GD building. The LMS and content were created, but the researcher soon realised that there had to be rules attached to it, to make the LMS acceptable and a quality tool worthwhile of motivating learners to access the LMS. The LMS was introduced using the "juggling" exercise, so learners were 'lured' into using the LMS to investigate the purpose in a latent manner. Using the LMS then became a "cool" activity, especially when learners started to show off their juggling skills on the playground, without even noticing that the researcher enforced APOS theory through juggling. The researched gained this experience during his involvement with gifted child education when he allowed learners to test their skills in extraordinary ways, for example, letting the children play "Double Dutch" (<https://www.youtube.com/watch?v=0a3jbQ5Edvo>) on the playground. Immediately, this type of behaviour attracts much attention and usually learners who were not noticed become instant celebrities overnight, boosting their urge to add to their academic skills.

Latent thus means that the intervention forced learners to visit the LMS in an implied manner in that learners are not forced or feel forced to use it. The LMS should not be subjected to any external influences such as power outages, which might negatively influence learners' access and impede learning when Moodle is inaccessible. During coding, learners could use Moodle to seek and find specific answers on problems they were confronted with. This was packaged in videos, or even in code-snippets, or as an explanation of various terminologies

(<http://wrru.co.za/moodle>). A strong point of the LMS was the clarity on assessments and tasks. The LMS gave an absolute guideline to learners as well as parents on what is expected. The cloud presence also tied in with their or the parents' email accounts so any tasks expected were transparent to all stakeholders. Learners raised an appreciation for being able to use the LMS to obtain clarity on a major assessment (Intervention 11B) which they had to prepare for.

6.2.1.7 Programming language theme

The programming language comprises 22 findings (Table 5.13). The Greenfoot programming language was the programming language of choice and it is the tool which contributes to computational thinking among learners. Pearson (2008) states that computational thinking requires more than basic literacy skills in IT and the Greenfoot programming language provides that incremental learning curve or development path from writing basic to complex applications (Intervention 12; Appendix N). Although Cohen (2013) states that learners using Alice also have a development path into Eclipse, the Greenfoot programming language is a one-stop shop with visuals and coding easily accessible by learners, as was the case with Intervention 11 (Appendix M), when learners cross-pollinated with image and sound utilities/generators in Windows. Java programming language statements can be used with the Greenfoot programming language. White and Sivitanides (2002) classify OOP in Figure 2.11, hence Java (Figure 5.29) as a productive and motivated language that may require both hemispheres of the brain. Any programming language relies on its users to know the IDE, and Greenfoot does have a visual interface that ties the coding with the IDE. Many languages used in education have a visual approach only, but mathematics also requires a language component (Tall, 2008), which the Greenfoot programming language provides as part of its syntactical component. This process can be found with many interventions in this research, such as when learners apply mental mechanisms, for example, reversal and encapsulation during the Object phase of APOS theory, applying the `Randomize()` method as an abstraction in the code.

The programming language, which also acts as a meta-cognitive system for this research, shows that learners, except one, were unfamiliar with any programming language at the time of this research. Being familiar with the Greenfoot programming language would have created some programming language belief system prior to this research, which would have been difficult to measure whether the belief system is positive or negative. This meta-belief system activity (MBSA) is built on meta-cognition as introduced by Flavell (1976) and transformed into a tangible framework by Moscucci (2007).

The Greenfoot programming language has a gradual progression in terms of levels of difficulty, which allows for a breadth-first hierarchical organisation (Zeitz & Spoehr, 1989) to become an expert in a programming language, as found during this research in section 5.4.3.19 supported by section 5.4 (Figure 5.35 to Figure 5.35). Through APOS it relates to mathematics (section 5.4.3.19) and is well documented on the internet as well in the Moodle LMS developed by the researcher (Intervention 5A; Appendix G). Research done by Brennan and Resnick (2012) as well as Papavlasopoulou, Giannkos and Jaccheri (2019) on Scratch as a block-based visual programming language for NPEs, allows learners and teachers to gain easier entry into the world of programming (Papadakis et al., 2014). The Greenfoot programming language is open source, providing dual-modality to learners, which does not only support the block-based, visual NPE approach, but also provide the gateway for learners to engage as programmers in a syntax-based programming language. Portnoff (2018) states that programming language education relates to learning a natural language, also supported by Tall (2008) that languages are paramount in mathematics learning. SA being a developing country of consumption, as found by Mpofu and Nicolaides (2019), is urged to embrace 4IR. Learners may partake in the Greenfoot programming language activity without costs in acquiring the software. It also provides a visual component that acts as a way of embodiment or enactment (Tall, 2008; section 5.2). The learner can envelop (make part of his/her body) him/herself in totality when using the Greenfoot programming language, i.e. becoming the Actor within the scenario, which is similar to mathematics learning (Tall 2008). This was observed during interventions when learners mimicked the Actor object by walking around the classroom, as if the classroom was the world within the Greenfoot programming language (Intervention 9; Appendix K). Other programming languages can be used, but the embodiment features, ease of use, and the progression from visual to syntactical programming are of importance not only to reach maturity in computational thinking, but also to support the 4IR envisaged by SA for all its learners. The flipped classroom examples in Moodle and the simple ways to create a scenario guarantee success, which builds confidence in learners. This also allowed metacognition (Flavell, 1976; Livingston, 2003; Jackson, 2004; Moritz & Lysaker, 2018) or the progression from the Action to Process phases much faster, and the programming language provided several ways that learners could investigate any problem during the Action phase to understand the challenge.

The Process phase of creating scenarios was achieved faster through using the LMS as reference tool for learners (Intervention 11). Difficult concepts such as dimensions in the Greenfoot programming language was achieved in the visual output of the two-dimensional interface. The programming language also possesses a debugging function, and difficult control structures such as the IF statement were broken down into Actions to simplify understanding. Even making the applications more natural, the Randomize function (section

5.4.3.21) was researched using the IDE and Actions and Processes. The programming language also consists of built-in functions or methods to allow abstraction as part of computational thinking. Once again, the learners needed to investigate these methods to understand what abstraction entails. The programming language allowed the creation of gaming applications with a naturalistic approach. A rocket intervention (14B) showed that it could perform certain actions that simulate reality in using control structures such as IF statements and variables, especially Boolean variables. Processes developed from these Actions led to an Object, such as method building to describe an abstracted concept (atWorldEdge()). These abstracted concepts contributed to the learners' Greenfoot Schema.

Many researchers (Brennan, 2012; Brennan & Resnick, 2012; Papavlasopoulou, Giannkos & Jaccheri, 2019; Yu & Roque, 2019) use Scratch as a favourite programming language to grow computational thinking among learners. However, debugging and use of language constructs are seen as important components of a programming language to assist learners with computational thinking, which is prominent in Greenfoot programming language.

6.2.1.8 Technical theme

The Technical theme comprises of four findings (Table 5.13). The technical platform is an important part of this research, as it influences the performance of learners. The Moodle LMS should not be influenced externally that which dictates the actual presentation of a class or lesson. Any school may construct an inexpensive PC from scrap parts to drive Moodle and the Greenfoot programming language as seen in Intervention 4A where the researcher used a Dual Core Pentium with 4 Gigabyte of memory. Within the classroom, PCs should be the option compared to RDP, to run Greenfoot, as it was a challenge during this research to work on terminals; it hugely impacted the speed of teaching and led to learner frustration. Greenfoot uses Java and needs to compile the scenarios prior to execution, which means that learners will have to wait for compilation to complete on the server. The RDP is a shared environment which changes the PC into a terminal or in the research environment the private school invested in terminals only. Learner logins may go into a queue when processing is requested for any scenario. The time to service the queue may cause frustration among learners due to learners having to wait for compilation to take place. With the LMS in the cloud (Intervention 4B; Appendix F-2), access is ubiquitous and separate from the compiling challenge, which alleviates requests to the same server platform. The maintenance and worry of a hard drive crashing is also minimised. If a school runs the LMS from a local PC as part of their network, the hard drive management must have a failover to avoid any breakdown in classes. This researcher did not spend money on failovers and lost the physical hard drive in the process, although it was an enterprise drive. The hard drive lasted for two years, which is quite good when even a class of 50 WCED teachers at two

conferences logged in simultaneously. The hard drive was also misused by teacher classes presented on behalf of Oracle and learner access 24/7. The domain changed from <http://efundo.co.za> to <http://wrru.co.za/moodle> depicted in Appendix F-2.

From an academic and educational perspective, the researcher used the TPACK (Koehler & Mishra, 2009) framework. The TPACK framework provides a shift from WHAT to HOW. These domains (TPACK) cannot live in isolation and need to be integrated as illustrated in [Figure 2.19](#). These three domains are interconnecting as TCK, TPK, PCK and the intersection of TPACK. Overall, the augmented TPACK is just a framework and needs a specific conceptual strategy from a research perspective, but is mandatory for this research to be considered by teachers and researchers prior to rollout.

The original PCK framework focuses on “why the high school learner should learn to program” as opposed to “why learners should be taught programming at all”. The complexity of adding a visual programming environment into the mix makes manipulation of IT related content implicit. The TPACK (Koehler & Mishra, 2009) would thus provide a better focus on IT and specifically the programming mix within the PCK mix. The focus is on how computational thinking is promoted when using programming concepts and APOS theory as lens, with a constructionist approach.

The TPACK of programming (Koehler & Mishra, 2009:67) focus on the thought processes and knowledge of teachers and the actions taken that may lead to observable effects. The thought processes translate to “why teach a programming language”, “what to teach”, “what are the problems in learning a programming language” and then decide on “how a programming language should be taught” which may lead to observable effects by focusing on the structure/syntax of the programming language, not paying too much attention to solving problems. Developing algorithms to solve problems are necessary to engage learners in using APOS theory within the programming language to achieve computational thinking (Cetin & Dubinsky, 2017), it cannot be separated from the physical technical platform. Leaving the problem-solving component to the next level, may remove the very essence of the advantage in teaching a programming language in the first place. Complexity of syntax must thus be minimised or even removed altogether, so that the learner only focuses on problem solving, gathering implicit syntactical knowledge also known as tacit knowledge.

6.2.2 Interviews

The interviews were conducted on the impact of Greenfoot programming language and mathematics on learners. The objective was to examine if the learners interpreted mathematics in a different manner, as they do with a programming language and if they

could relate Greenfoot programming language with mathematics learning. These interviews (Appendix R-1, R-2 and R-3) comprise of an algebra exercise on simplification and a science assessment on the electrical circuit. These examples were model examples of what the learners were busy with at the time the research happened, so they can relate the theory to that of a programming language. The interviews conducted led to 12 findings (Table 5.15). The five themes that crystallised from the categories were APOS, Beliefs, Cognitive balance, Computational Thinking and Learning. These themes are now discussed in the next sections.

6.2.2.1 APOS

Although learners were cognisant of the programming language and how it related to APOS theory, they were unable to advance from one phase to another in mathematics. They knew what and how to do it using a programming language and they knew what question to ask in mathematics, but needed a teacher or mediator to provide answers to their questions. The mechanics of what to do in mathematics were absent, and although they knew that, for example, an Action needed to be taken, they could not figure out what Action to take. They could not explain or interpret such Actions, although they identified that some Action is needed. The researcher fulfilled the role of the teacher during these sessions. Only when the researcher guided them, did they realise the potential of APOS theory. Learners followed rote learning to fulfill mathematical concept images of mathematical definitions, which became concept images instead of concept definitions. The learners were more interested in the answer than the logic by memorising the drawing of the triangle for the equation $V=IR$, part of the “electrical current” exercise. The emphasis was on the physical drawing and not on the understanding of the equation. The problem was that the concept image satisfied the need to generate answers and obtain marks for the assessments. Learners were more concerned about the formula and got lost in silos of concept images. These activities did not complement nor enabled computational thinking and thus did not promote APOS. When the questions read “Determine Voltage”, they read the V off the formula image triangle and used a calculator to calculate the answer. Although the CAPS document stated that learners should resort to mental calculations, learners adhered to the initial directive of using calculators. Only when the researcher guided these learners and broke down the problems into actionable steps, could they understand that only $V=IR$ is needed. However, they still found it difficult to identify these steps on their own. The learners now realised what questions to ask the researcher to ensure understanding of mathematical concept definitions. APOS theory provided learners with the skill to ask questions about their problem in mathematics. They wanted to take Action to get to the Object phase of the mathematical concept, but needed someone to provide those answers.

6.2.2.2 Beliefs

Beliefs about mathematics were shaped through “met-befores” that built up through the years. These caused incorrect answers because actual Schemas did not exist. The learners who were interviewed became anxious when the researcher asked them questions on mathematics. Although the programming language was a new belief that took on shape, mathematics was part of an existing belief system and sometimes cluttered the learners’ understanding of programming language concepts. The anxiety showed that their confidence levels were low, even before the question was posed.

6.2.2.3 Cognitive Balance

Learners were unable to abstract in general. Greenfoot was a challenge for some learners. Balancing all aspects of Greenfoot in what should be memorised and what not, were realised, but for mathematics it was unclear. Cognitive load theory seemed to be absent. The researcher tried to understand how learners’ cognitive load was minimised to turn mathematics around into metacognition, but this will be recommended as further research. A programming language necessitates that the learner understands the entire Schema on the Greenfoot programming language. In mathematics, learners managed rote learning by memorising silos of mathematics concepts such as the $V=IR$ example (section 2.3).

6.2.2.4 Computational Thinking

Computational thinking became a challenge when learners could not reflect on mathematics as they managed to do for a programming language. The abstraction challenges were problematic, as the researcher had to guide them in making the association. Schemas were also absent in mathematics, which hindered computational thinking within mathematics due to silos of concept images. These concept images allowed learners to gain marks pertaining to that concept image, but the cognitive load became heavier (section 5.2).

6.2.2.5 Learning

Learners were influenced by teachers who forced learners not to use actions as part of their discovery because it was supposed not to happen at this stage of their lives. Learners complained, when the researcher showed the steps of a problem, that teachers do not want them to write down these steps as it was assumed that they must have the ability to do mental calculation, as prescribed in the CAPS document. The problem with that was that some learners never crossed the Action phase on the concept definitions and simply could not continue to the Process phase to reach the Object phase and to scaffold to their Schemas. The result was that learners simply followed any route that seemed a possible solution. Learners depended strongly on the pyramid that housed the formula for $V=IR$, and without that recipe, their answers were wrong and they failed to think beyond the pyramid.

Figure 2.13 is a typical example where an action could have saved learners from memorising silos of concept images.

6.3 Summary

This research afforded meta-learning opportunities to learners through APOS, Beliefs, Cognitive Balance, Computational Thinking, LMS, Learning, Programming Language and Technical themes, which emerged from the findings.

Replicating the research, APOS theory as the epicentre of this research needs to be introduced to learners through meta-cognitive processes with the proposed conceptual framework as lens. The proposed conceptual framework together with GDs through ACE should act as guidelines towards introducing computational thinking through APOS theory within a programming language. The APOS theme shows that learners need a hands-on approach to let learners gain a practical understanding of APOS theory prior to venturing into the Greenfoot programming language. Learners need to believe in APOS theory through practical activities such as juggling and a PL. Learners are enveloped by mental mechanisms when practicing APOS theory during programming while algorithms are produced to solve problems, as learners move through these mental structures. The Beliefs theme shows that the beliefs of learners may influence their attitudes when dealing with this wicked problem of computational thinking. Domains such as mathematics and programming that needs learners to perform at a cognitive level of formal operations need a specific didactical approach which will ensure a positive outcome in a learner's education. The Cognitive Balance theme shows that cognitive load theory must be included in the proposed conceptual framework, to ensure learners taking on more academic tasks, but delivering the same or better performance. The Computational Thinking theme emphasises abstraction, automation and thought processes, which are embedded in practising the Greenfoot programming language. The Learning theme shows that learners have more confidence in their learning through cognitive load theory and a supporting LMS that balances learner working memory. The LMS embedded within the LMS theme provides ubiquitous behaviour of academic support. The Programming Language/Coding theme shows the literature study supports Greenfoot as a language that provides opportunity to learners to make progress from the visual to the syntactical approaches in writing algorithms. The Technical theme shows the dependency on hardware platforms to make or break the research. Disadvantaged communities and schools can also reap the benefits of using older PCs to act as a research platform.

All these themes are interrelated, providing a secure didactic platform where computational thinking can be developed and practised. The interviews highlighted APOS, Beliefs, Cognitive Balance, Computational Thinking and Learning themes. Within the APOS theme,

learners could form an analogy between mathematics and programming and knew what questions to ask based on APOS theory, but needed a teacher to explain those questions to them. The Beliefs theme shows that “met-befores” are crucial and mathematics had more casualties than success story learners. It is also not always true that a learner enjoys mathematics if he/she performs well. The anxiety that comes with the exercise can be enormous. The Cognitive Balance theme shows that learners’ cognitive load in mathematics was not built on cognitive load theory, as was the case with the Greenfoot programming language that was rolled out in a structured manner, supported by an LMS to enforce cognitive load theory. The Computational Thinking theme shows that computational thinking was a challenge when learners could not reflect on mathematics, as was the case with the Greenfoot programming language. Computational thinking in mathematics was built on concept images more often than not. The Learning theme shows that learners did not follow APOS theory in mathematics as with the Greenfoot programming language, where their answers in mathematics were based on either guesses or concept images, i.e. silos of ideas about mathematical concepts and not per concept definition.

The interventions executed with this research communicated the concept of academic choice to learners. Within the proposed conceptual framework, learners were confident and honest about the choice they had to make to achieve success.

CHAPTER 7: CONCLUSION AND RECOMMENDATIONS

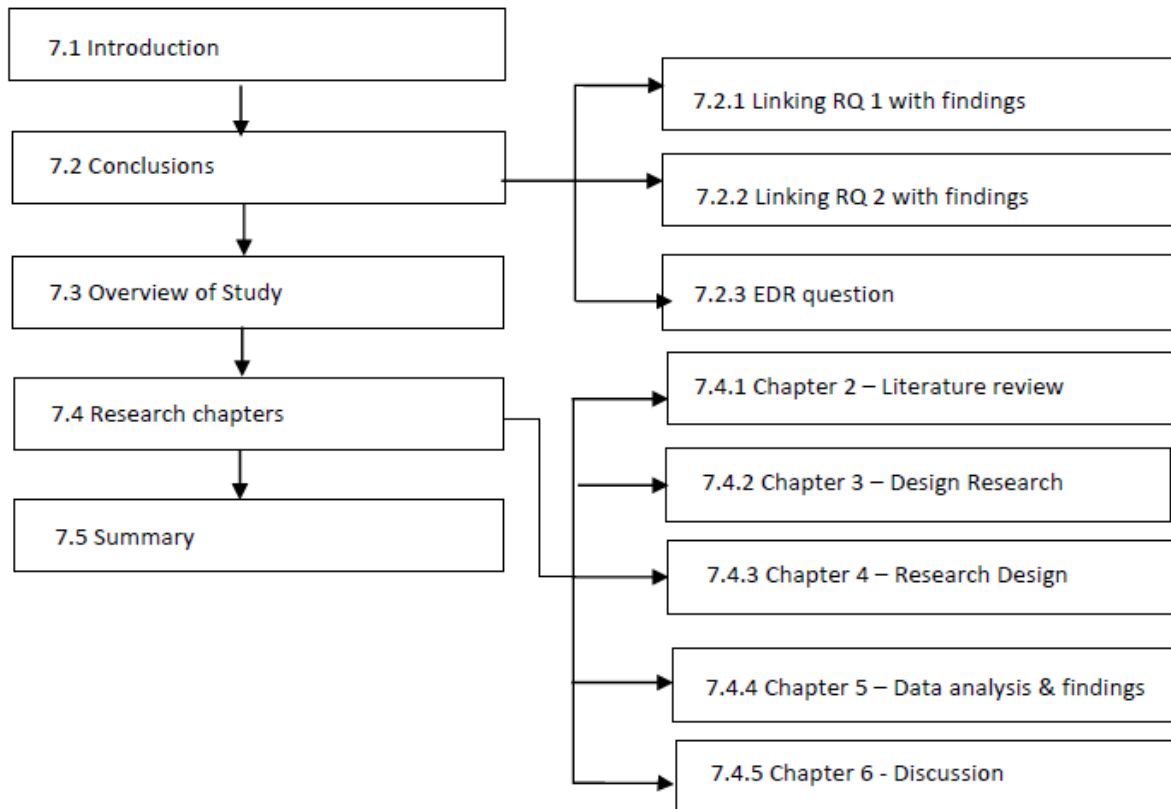


Figure 7.1: Chapter layout

7.1 Introduction

In this Chapter, [Figure 7.1](#) shows the layout of this Chapter. Conclusions are drawn on the role of programming language concepts and APOS theory in order to promote computational thinking. The problem statement of this research states that it is not clear how high school learners' computational thinking may be promoted at a cognitive level of formal operations. This wicked problem is explored through two research questions namely, "What are the characteristics of an enhanced learner's teaching and learning strategy that can empower learners to master computational thinking skills through APOS theory, infused by a programming language at high school level?" and "How can computational thinking skills at a cognitive level of formal operations be promoted among high school learners through the teaching of a programming language aligned to Action Process Object Schema (APOS)?" The findings are linked to the research questions in this Chapter.

It is concluded that by using a programming language, it may promote computational thinking among learners using APOS theory as lens. Furthermore, mathematics must be revisited by the teachers of mathematics in addressing the learners' questions based on their

programming language experience. By using a programming language, learners know what and how to use APOS theory to approach their misunderstanding of mathematical concepts, using the proposed conceptual framework. The APOS theory gives guidance to learners to understand how APOS theory can enrich learners' cognitive stance through meta-learning. Chapter 7 consists of (i) sections, (ii) conclusions, (iii) and recommendations ([Figure 7.1](#)).

7.2 Conclusions

High school learners underperform at a cognitive level of formal operations when engaging in subjects such as Mathematics and Science. Where computational thinking is about abstraction, automation, thought processes and decomposition, mathematical thinking concerns abstract structures and hence mathematical thinking is embedded in computational thinking. The problem statement of this research states that it is unclear how high school learners' computational thinking may be promoted at the cognitive level of formal operations. This wicked problem is investigated through two questions namely, "What are the characteristics of an enhanced learner's teaching and learning strategy that can empower learners to master computational thinking skills through APOS theory, infused by a programming language at high school level?" and "How can computational thinking skills at a cognitive level of formal operations be promoted among high school learners through the teaching of a programming language aligned to Action Process Object Schema (APOS)?".

This research used a programming language as a metacognitive system. The findings show that although a programming language paves the way to computational thinking, learners have set ways in mathematics and need the researcher to assist learners with remedial work, based on APOS theory questions. The researcher had to answer their questions which emerged from their programming language approaches and experiences of computational thinking when applying APOS theory within the Greenfoot programming language. The learners now have a frame of reference to engage in computational thinking when solving a problem.

The programming language is not a magic wand to provide solutions to all the learners' questions, but at it least empowered them to know which questions to ask and what actions to take to remedy a lost concept definition in mathematics or to avoid the stacking of concept images. The programming language provided learners with a blue print on how to approach their mathematics problems and with a choice to be successful.

The next section shows how the research addresses the research questions ([Table 5.14](#)).

7.2.1 Linking RQ 1 with the findings

Refer to [Figure 5.14](#) to see the mapping of the findings to the themes per research question. Each research question is now discussed.

RQ 1: What are the characteristics of an enhanced learner's teaching and learning strategy that can empower learners to master computational thinking skills through APOS theory, infused by a programming language at high school level?

The characteristics of an enhanced learner's Teaching and Learning (T&L) strategy, which focus on computational thinking, are prominent in the *computational thinking*, *programming language*, *APOS* and *technical* themes, as discussed in the SRQs below.

SRQ 1.1: What factors are needed to promote computational thinking at a cognitive level of formal operations among high school learners?

The themes that were prominent during the exploration of this question include *computational thinking*, *programming language* and *technical*. The interventions show that abstraction is a problem for 50% of the learners in this research. During the literature study the focus was put on PLs, based on the views of Wing (2008), Aho (2012) and Denning (2017), who advocate a strong link to Computer Science and Programming, and more specific, as the automation part of computational thinking. A programming language such as Greenfoot does provide the visual and syntactical approaches learners and teachers may explore to build computational thinking through thought processes (Selby & Woollard, 2014). Assessments and questionnaires supported the validity and rigour of the EDR process. Mental mechanisms such as interiorisation, encapsulation, de-encapsulation, coordination, reversal, generalisation and thematisation are practiced from within Greenfoot as learners illustrated mechanisms these during the interventions to align reflective abstraction within APOS theory with computational thinking (Cetin & Dubinsky, 2017).

Teachers may either use the examples as performed in this research or use examples that focus on any of the mental mechanisms for specific APOS mental constructions. These mental mechanisms within APOS align reflective abstraction and computational thinking, which connects computational thinking, mathematics and the programming language when building Schemas. Computational thinking consists of automation and abstraction, and the automation component is found in the programming language as advocated by Wing (2008, 2011), Aho (2012) and (Denning 2017). Embodiment (Tall, 2004, 2008) is also an important component during learners' development in mathematics and the Greenfoot programming language strongly advocates embodiment, as summarised in the findings during the enactment of problems by learners when using Greenfoot. All of the above as summarised

as thought processes which need to be stimulated, as promoted by Aho (2012), Selby and Woollard (2014), Cetin and Dubinsky (2017) and Denning (2017). The answer to the SRQ is that the factors needed for the development of computational thinking at a cognitive level of formal operations among high school learners are abstraction, automation, mental structures, mental mechanisms and a programming language necessary to develop computational thinking among learners.

SRQ 1.2: What type of programming language may be used to promote computational thinking skills at a cognitive level of formal operations?

Greenfoot programming language, as guided by the literature study on PLs, was chosen as the programming language for this study and supports the automation component as advocated by Wing (2008). The selected programming language must allow learners to operate at a cognitive level of formal operations by providing a progression path from the visual to syntactical approaches. The visual part of Greenfoot provides learners with the functionality to perform mental mechanisms by constructing an algorithm to solve the problem. Learners can see the result of the visual approach in the instructions or code, which provides immediate feedback on their proposed algorithm. The visual approach aligns to syntax, and *vice versa*. Greenfoot provides a constructionist approach, where learners learn through the construction of an application or an algorithm in Greenfoot, while at the same time adhering to cognitive load theory through an LMS (Papert, 1980). Although the programming language was unknown to the learners, Greenfoot has a visual interface that enables learners to interact without having to program at first, as done during the first number of interventions and hence the availability of a progression path. The focus is on solving the problem rather than worrying about syntax. Syntax is a challenge for most learners when involved in programming and it discourages many from pursuing programming (Malan & Leitner, 2007; Saeli et al., 2011). Greenfoot provides a progression path that allows learners to climb the ladder of complexity to become involved in mental mechanisms by developing methods to hold complex coding. The interventions as described in Chapters 4 and 5 adopted this gradual approach. This allowed learners to apply abstraction, as shown in this research. The visual characteristics and progression path if Greenfoot affords its users support with developing of computational thinking through a constructionist approach. Having stated all of the above, learners became irritated when the Greenfoot programming language took 5 minutes to show after compilation. The delay has led to learners performing unnecessary additional clicks using the mouse. An additional burden was place on the CPU that processd on the server, which delayed the compilation process even further. Considering the TPACK framework (Koehler & Mishra, 2009), the technical platform does play an important role when lessons are rolled out. Implementing Moodle and controlling

sequential compilation temporarily alleviated the compilation problem, and LMS searches directed learners to the Internet. The learners and researcher could tolerate the situation, which was not an ideal didactic environment. Greenfoot thus fits into the breadth-first hierarchical organisation of learners for them to become experts (Zeitz & Spoehr, 1989). The Greenfoot programming language offers more than the block-based languages, as promoted by Brennan and Resnick (2012) and Papavlasopoulou, Giannkos and Jaccheri (2019). Greenfoot programming language is open source and offers dual-modality to learners, which entails a visual NPE approach and provides a gateway for learners to engage as programmer in a syntax-based programming language, as advocated by Portnoff (2018) in that learning to program also entails a natural language learning approach.

SRQ 1.3: What constructs within the programming language facilitate APOS theory at a cognitive level of formal operations?

Greenfoot, as many other languages, comprises constructs such as control structures found in loops and conditional statements. The IDE is built into Greenfoot with a “help” option that describes the different classes and methods supported by examples for learners. Using the control structures and IDE, learners are forced into thought processes about a problem or an algorithm on the problem. The outcome of implementing these control structures is visible to learners during compilation and execution, which means that learners are already able to evaluate their outcomes as successful or not during these stages. Learners can see their algorithms as outcome in visual format. Denning (2017) notes correctly that the executing of applications alone does not turn a learner into a computational thinker, but discovering algorithms does. This is an immediate outcome compared to mathematics where learners cannot determine the correctness through inspection; neither may their peers ask to evaluate the outcome of a mathematical problem. Constructs such as the IF statement, methods, looping structures, and variables, together with the IDE, facilitate APOS theory in mental mechanisms for mental constructions.

7.2.2 Linking RQ 2 with the findings

RQ 2: How can computational thinking skills at a cognitive level of formal operations be promoted among high school learners through the teaching of a programming language aligned to Action Process Object Schema (APOS)?

In answering this research question, programming language, Cognitive Balance, Moodle (LMS) and Learning were prominent themes will be discussed when answering the following SRQs.

SRQ 2.1: How are the constructs of a programming language taught among high school learners at a cognitive level of formal operations?

As advocated by Denning (2017) in [Figure 4.11](#), Greenfoot as computational notation is used to compute some abstraction or computational model, which is controlled by an algorithm developed by the learner. In developing the algorithm and mapping it into Greenfoot, thought processes are inevitable and the learners engage in computational thinking as guided by APOS theory. This, however, does have an effect on learners being mentally overloaded with new concepts and ways to be creative in their thought processes when adhering to the traditional didactics. The findings linked to this question include 2-3, 3C-2, 3C-3, 5A-1, 7-1, 7-2, 8-2 and 14B-2, depicted in [Table 5.14](#). The T&L strategy dictates interventions, and this has led to these findings. The Greenfoot programming language was installed on a PC and supported by an LMS to guide and assist learners with obtaining answers and ensuring a cognitive balance through cognitive load theory to fulfil these tasks. Assignments are given to learners to engage with the Greenfoot programming language in solving problems by developing algorithms for such problems. The assignments can be grounded on GDs with ACE (Intervention 2A-2; [Table 2.1](#)) to assist learners with following the correct steps when developing algorithms. The mental mechanisms triggered by APOS theory (mental structures) must be highlighted to learners for them to make any assignments part of a constructionist approach governed by computational thinking through the APOS theory lens.

SRQ 2.2: How do the constructs of a programming language align to APOS among high school learners at a cognitive level of formal operations?

As with many other languages, Greenfoot does have a debugging option. Learners are taught to add comments to code to assist them with tracing the output of their code. The visual interface of Greenfoot also allows learners to see what the actor does when the code executes. The programming language theme was prominent during findings 2-3, 8-5, 9-2, 9-3, 9-4, 9-5, 10-1, 10-2, 10-3, 11B-3, 12-1, 12-2, 12-3, 12-5, 13-3, 14B-2, 14D-1 and 14D-3. The control structures mainly consist of conditional and repetitive constructs such as IF and “for” looping statements, which the researcher crafted into interventions that forced learners to develop an algorithm. Intervention 13 was such an intervention that introduced a Greenfoot variable to enable a balloon actor instance to move horizontally and vertically. The term ‘instance’ is specifically used here, because the learner may generate a bouquet of balloon instances by repetitively creating a new balloon Actor object. Learners had to consider a grid with an x and y axis and brainstorm the problem to move these balloons in any direction. Having constructed the `atWorldEdge()` ([Figure 5.32](#)) method in a constructionist manner a new dimension was introduced, where learners had to consider how this object could be used within the current scenario. The method `atWorldEdge()` is

regarded as an Object that forms part of the code to verify if the Actor object will collide with the edges. The method name `atWorldEdge()` is also used throughout the thesis, so the reader may relate to terminology such as method, encapsulation, reversal, other mental mechanisms, syntax and so on. These method names vary from learner to learner, seeing that it is a personal creation with different code structures, depending on the aim of the method. Normal coding rules such as Pascal case was adhered to, hence the mixing of upper- and lower-case letters that form the method or variable names. It plays the role of an Object because it is reusable code, not only within the same scenario when used by other actors as well, but also across scenarios. The learners had to control different colour balloons that drifted horizontally and vertically, which forced the learner to check if the left or right margin was reached. Upon detecting these sides, the balloon changed direction. This called for an IF statement. The for-loop is implied in that the run command calls the `act()` method within the scenario repetitively. These constructs are debugged so learners can trace their instructions that serviced the algorithm in the Act button. The steps are executed per line and the learners can compare the code to the outcome or output visually step-by-step. In some cases, learners only performed physical actions such as walking inside the classroom, simulating the movement of the balloons. This is where the learner became the Actor object and partook in an activity called enacting, which is also part of mathematical thinking. APOS theory is followed by learners being in an Action, Process or Object phase; an object such as `atWorldEdge` is broken down into actions again to discover a better Object that will fit the purpose to solve the problem. If learners investigate the (APOS) Object, `atWorldEdge`, they can de-encapsulate the Object through actions. If the code embedded within the method becomes unclear or do not suit the purpose of the method, learners use specific mental mechanisms to decompose the Object and compose a better Object. The reason can be code that is not generalised enough, or components or chunks of code influenced by cohesion. The whole process of interiorisation (mental mechanism, section 2.2.2.1(c)(vii), Figure 2.17) then restarts as learners configure a new Process within their minds to structure a method through encapsulation (Figure 2.17). Actions are imposed on Objects, leading to a Process. Learners can generalise these encapsulated methods as re-usable code. Overall, a Schema exists within the learner's mind on how to create a scenario that will allow balloons to move in a direction using an Object such as `atWorldEdge()` method, which is reusable to solve other problems through programming. Through scaffolding, the Schema was strengthened. Learners cannot build silos of concept images as with mathematics, because it cannot suffice in coding. Coding expects learners to link constructs which cannot exist as silos. This Schema is thus a dynamic entity on which actions are imposed, that may lead to new Processes and Objects in APOS theory. This is what Piaget (1977) describes as assimilation, when actions become thematised objects or accommodation (Figure 1.2)

depending on the status or relationship of the learner with the programming language and its structures.

SRQ 2.3: How does the use of an LMS, as a platform for learning, aid the teaching of a programming language aligned to APOS to promote computational thinking skills at a cognitive level of formal operations among high school learners?

The themes that emerged are the Cognitive Balance theme in findings 2A-1, 2C-1, 2C-2, 14C-2, 14C-2, the *LMS* theme found within findings 2B-1, 2B-2, 3A-1, 3B-5, 4B-1, 4B-2, 4B-3, 5B-1, 5B-2 and the *learning* theme in finding 11A-2. Learners want to obtain solutions for problems, but the coding world is just too broad to pursue as an individual. Learners show interest in having Moodle as the LMS to watch a video on coding to discover a possible way to solve a problem. The learners found the extra academic burden as doable because cognitive balance was implemented through the LMS and assistance was provided by the researcher as Oracle instructor. This is necessary, as learners rather need to focus on creativity than worrying about where they will find answers to these problems. Tall (2008) argues that the human brain is very limited and only deals with a small number of pieces of information at a time. The Moodle LMS gives learners the opportunity to discover new knowledge on the Greenfoot programming language, such as how to compose artwork in creating World or Actor objects through Paint and recording sound clips associated with these Actors. At the same time, the LMS acts as a repository for elaborate explanations that need to be synthesised by the learner. The researcher assessed learners' practically and gave the learners a questionnaire afterwards to assess their knowledge on the coding of certain constructs. The learners answered these questions with meticulous descriptions about the processes through code snippets, especially for Intervention 7 (Figure 5.33). Moodle acts as a repository where specific answers could be found to assist learners in their tasks, as moulded by the researcher as instructional designer. This also alleviates the burden of memorising too many facts in the learner's working memory. When learners were exposed to Moodle, their creativity triggered, as found in the enthusiasm with which learners participated, especially during assessments.

7.2.3 EDR Question

The EDR question is answered above, namely: "What teaching and learning strategies can empower learners to master computational thinking skills, through APOS theory, which is expected to function at Piaget's (1977) cognitive level of formal operations, infused by concepts and characteristics of a programming language at high schools, in order to cope with the challenges in subjects such as Mathematics and Science?"

The EDR question combines both research questions as discussed above to simplify the goals by the van Wyk and de Villiers (2018) EDR model. The teaching and learning strategy comprises the ACE framework (Figure 2.20) of Dubinsky and McDonald (2001), tied to the GD, which involves Greenfoot as programming language. Moodle LMS provides a cognitive balance to let learners solve problems through Greenfoot with a computational thinking focus. The cognitive level of Piaget (1977) was adhered to by starting this research with grade 8 learners to tap into their cognitive level of formal operations. The strategy is further strengthened by a technical platform that provides Greenfoot programming language learners building their computational thinking skills by referring to the TPACK framework of Koehler and Mishra (2009), who enhanced the Shulman (1986) PCK framework.

Mathematics was not realised as a spin-off in this research, but with computational thinking as the focus, learners could criticise their mathematical problems based on APOS theory. Learners demonstrated, during the interviews, what steps to take when confronted by inconsistent understanding of mathematics concept definitions. Teachers in mathematics need to assist these learners with unlocking those concept images and changing it to concept definitions through the learners' ability to ask better questions. Learners thus do relate to their experiences in Greenfoot, moving hence and forth within APOS theory's mental constructions and mental mechanisms. However, the status of the didactical contract that exists between teacher and learner dictates how teachers will respond to learners newly acquired understanding of computational thinking in mathematics. It may add additional strain on the approach of teachers to these new way/didactics of addressing mathematics, influencing the success of such a rollout.

7.3 Overview of the Study

The aim of this research was to explore and understand how a programming language, using Action Process Object Schema (APOS) theory as lens, could promote computational thinking skills at a cognitive level of formal operations among high school learners. The study was conducted at a private high school in the Western Cape.

The research methodology was based on an interpretivist research philosophy. The ontological underpinning of the study was subjective, and the epistemological stance accepted opinions of learners through written, spoken and visually attributed meanings. The axiology of the researcher was that of a practising educator in programming and industry, as teaching and learning expert, and as a certified Java-Greenfoot instructor through Oracle.

Data were collected during lectures, through observations, interviews, assignments and assessments using the EDR approach. Using Greenfoot as a programming language, supported by Moodle as Learner Management System (LMS), learners discovered

programming through “worked examples” and a constructionist approach as proposed by Papert (2005). The terminologies ‘constructivist’ and ‘constructionist’ are viewed by many researchers as the same, as indicated to the researcher by Prof Dubinsky (2015) in an email, when Prof Dubinsky was asked if he distinguishes between constructivism and constructionism. For the purpose of this research, the definition of Papert (1980) was adhered to. Qualitative data analysis was done through data condensation, data display and drawing and verification of conclusions using thematic analysis.

The research Chapters 2 to 6 are summarised in the following sections.

7.4 Research Chapters

7.4.1 Chapter 2 – Literature review

The literature study followed the hermeneutic framework of Boell and Cecez-Kecmanovic (2014:264) (illustrated in [Figure 1.2](#), section 1.8). The literature study cast light on discovering a framework that promotes mathematical problem-solving approaches. This led to prominent concepts within mathematical thinking such as abstraction, APOS theory, computational thinking, constructivist and constructionist approaches and different types of epistemologies that may enlighten and support the relationship between mathematical thinking and computational thinking. Through the literature study, RQ 1, “What are the characteristics of an enhanced learner’s teaching and learning strategy that can empower learners to master computational thinking skills through APOS theory, infused by a programming language at high school level?” was answered. The research question was broken down into factors necessary for computational thinking, type of programming language necessary to promote computational thinking, and the constructs within such a programming language to facilitate APOS theory (section 2.2.2).

The literature study gave more clarity on Piaget’s (1977) stages of development to classify the sample group of learners into a specific age and grade, also known as the level of formal operations that fit the profile of grade 8 and 9 learners (section 2.2.5). Further reading and analysis showed that mathematical learning revolves around embodiment, which ties in with APOS theory during mathematical learning. The status of outcomes-based education (OBE) raised concerns in exposing constructivist approaches as a constructivist teaching fallacy and how this may impede learning among learners. The “worked-example effect” was highlighted and accepted into this research as a more appropriate learning technique strengthening the constructionist approach.

The relationship between computational thinking, mathematical problem-solving approaches and a programming language aligned well with the three worlds of mathematics. Barrouillet (2015) also distinguished actions from gestures, which highlighted a concern in Piaget’s

(1977) theories, but serves for future research especially when looking at teaching and learning among deaf and blind learners.

The choice of a programming language was Greenfoot. Compared to other languages, Greenfoot supports a progression path for learners in the form of a graphical user interface through to the usage of language constructs with syntax and debugging properties. Greenfoot was built on Java, and Java constructs can be used directly within Greenfoot. Greenfoot is a gaming language which supports APOS theory when developing algorithms for problems as investigated in this research. The progression path allows learners to continue their programming experience in developing enterprise solutions and not having to implement a drastic change in their programming skills by changing the programming language. A most viable framework in teaching and learning is built on the restructured pedagogical content knowledge or technical pedagogical and content framework which recognises IT as part of the teaching and learning process recognised within this research.

The literacy framework of Prensky (2008) is now changed to include computational thinking as opposed to programming in accordance with Wing's (2006, 2008, 2011) proposal. Programming will be implied as part of computational thinking as highlighted in this research. The APOS mental constructions and mental mechanisms tie reflective abstraction and computational thinking together, which shows the connection between computational thinking, mathematics and the Greenfoot programming language, as illustrated in [Figure 2.21](#) when building Schemas. The final motivation why a programming language should be used is highlighted in the literature study through meta-cognitive systems. This research studied the promotion of computational thinking from an unknown perspective (programming language belief system) such as the Greenfoot programming language, which should act as a meta-cognitive approach, seeing that only one learner has encountered a programming language before in this research.

7.4.2 Chapter 3 – Design Research

Design Research (DR) stems from Design Science (DS), which is applied DS, also known as DSR. The DS should produce shareable theories to provide communication to practitioners and educational designers. Within the educational world, DR is also known as DBR or EDR. Takeda et al. (1990) regard cognition in DSR through abduction, deduction and circumscription as cognitive processes. This research used DBR as a validation study where a theory such as APOS was investigated and explored. There are many approaches in EDR in models and frameworks, but the van Wyk and de Villiers (2018) model was chosen for this research. An EDR question was formulated based on the van den Akker (1999) principle namely: "What teaching and learning strategies can empower learners to master

computational thinking skills, through APOS theory, which is expected to function at Piaget's (1977) cognitive level of formal operations, infused by concepts and characteristics of a programming language at high schools, in order to cope with the challenges in subjects such as Mathematics and Science?"

7.4.3 Chapter 4 – Research Design

Chapter 4 describes the research design, which started with Action Research, but changed to educational DR as determined by the outcomes of this research. The motivation for EDR was the artefacts as outcome, computational thinking as the wicked problem to be researched (section 3.3), and a bevy of interventions that necessitated a move towards solving the wicked problem. The wicked problem is found in the problem statement. The EDR structure used design as promoted by the van Wyk and de Villiers (2018) model, based on interventions. The sample consisted of grade 8 and 9 learners sequentially. Sequentially means that the grade 8 learners continued with the research in grade 9. This research explored the role of computational thinking among high school learners and how computational thinking could be promoted in these learners. The following strategies were considered: (i) a demonstration case; (ii) Action research; (iii) DR; (iv) DSR; (v) EDR; (vi) interviews; and (vii) observations/ reflections. The DR strategy was DR (Figure 4.9), also known as EDR in education. The abstracted phases found in the van Wyk and de Villiers (2018) model are: (i) a preliminary research phase; (ii) the prototyping phase; and (iii) an assessment phase. The DR strategy is DR as strategy of choice (section 4.5.3).

7.4.4 Chapter 5 – Data analysis and findings

As the research explored the impact of computational thinking using a programming language with APOS theory as lens, the researcher focuses on the mathematical belief system and how it influences learners' attitudes towards mathematics (Moscucci, 2007) (section 5.2). This triggered the idea to use a programming language as a meta-cognitive approach and build a "clean" programming language belief system among learners to augment their beliefs about mathematics. Augment means that an analogy can be drawn by learners in comparing their existing system of belief about mathematics to that of the programming language belief system. The concept of mirror neurons is something that the researcher had to take cognisance of to build a positive belief system different from their current system of belief about mathematics. While structuring the interventions, care was taken to present the interventions in such a manner that it was didactically sound. This was based on a didactical contract that created positive experiences. Every time the researcher visited mathematics in the form of current problems done in class for learners to reflect, learners demonstrated silos of concept images in terms of mathematics. Further data collection processes and the analysis of data were covered. A qualitative design was used

through abduction on data that were gathered through interventions. Each intervention produced findings which were summarised into 8 themes.

7.4.5 Chapter 6 – Discussion

A total of eight themes were constructed, namely APOS, Beliefs, Cognitive Balance, Computational Thinking, LMS, Learning, Programming Language, and Technical. The van Wyk and de Villiers (2018) model proposes an evaluation and reflection stage. The reflection stage was expanded by using the framework of Gregor, Müller and Seidel (2013). The theoretical conceptual framework was accepted as a proposed conceptual framework in [Figure 8.4](#).

7.5 Summary

This Chapter highlighted the conclusions and recommendations of the research, which are discussed in the following sub-sections.

7.5.1 Conclusions

Using a dual-modality programming language may promote computational thinking among learners, but mathematics needs to be revisited by the teachers of mathematics in addressing the learners' questions based on their programming language experiences. Learners now know what and how to use and approach their misunderstanding of mathematical concepts, built on the conceptual framework involving a programming language, using APOS theory as lens. The actual guidance and coordination of mathematics subjects must stem from mathematics teachers. The APOS theory as lens guides learners to reflect on their education, using metacognition through the Greenfoot programming language practising computational thinking, compared to mathematical problem solving. All the research questions as well as the sub-research questions are linked to the findings within the research.

7.5.2 Recommendations

The following themes will be discussed that influenced the researcher's recommendations on replicating the research within education. The themes were arranged according to importance, which will secure success when rolled out, starting with the technical theme as the most important and concluding with a view on education recommendation in SA.

7.5.2.1 Programming language

NPEs in education comprise mainly of Scratch and App Inventor, with Scratch the most popular programming environment (Papadakis et al., 2014; Szabo et al., 2019). This research was pitched at a cognitive level of formal operations and points to an age group of

11 years and older. The NPE approach in using block-based programming will not suffice when writing code using APOS theory as lens to promote computational thinking. The programming language for this research accommodated a mixed modality, from visual programming to text-based programming. Learners that used the Greenfoot programming language enjoyed an easy start, using a frame-based visual approach to syntax-based interface.

7.5.2.2 Technical

This research should only be replicated among schools if the technical platform satisfies that of a PC environment and not a terminal environment. Servers supporting the IT infrastructure within a school need not use RDP technology, but may rely on basic PC environments for the Greenfoot programming language, which will increase the compilation and maintain learner attention. The PCs can be imaged using Norton image technology or Acronis technology where multiple PCs can be imaged or prepared on a network at the same time to minimise installation of software for each individual PC.

7.5.2.3 Cognitive Balance and Beliefs

A constructionist approach in using an LMS will promote cognitive load theory to maintain academic performance among learners, although the workload increases. The LMS should be developed by an instructional designer (ID), which can be configured by IDs within the WCED or knowledgeable IDs to ensure cognitive load theory and constructionism. The ID will structure the LMS content in such a way that it inhibits learning. The learner working memory (LWM) must be considered when evaluating the learner and school environment and the components of LWM must be discussed with all stakeholders to emphasise the roles of intrinsic and extrinsic/extraneous cognitive loads on learners. All stakeholders must be informed about the gravity of teaching programming and the impact of any new domain on future endeavours of these learners, seeing that it is a wicked problem being addressed.

7.5.2.4 LMS

The LMS is a decision that every school can make based on their finances available. Primarily, the IT teacher should involve IT skills of parents and install Centos Linux distro on a PC, but preferably on an old server donated to the school. The researcher specifically recommends Centos, as it is Windows based and is less cryptic compared to other distros. Always choose a stable version and not a Beta version. Remember we just want to run Moodle. Ensure the hard drives are of enterprise standard and if possible, enable RAID 1 using two hard drives if the hardware permits. The memory configuration should be 4 Gigs of Ram as a minimum requirement. Preparing bootup USB flash drives can be accomplished using Rufus, UNetbootin, win32diskmanager or balenaEtcher technologies. Normal DVD

drives have become too small, seeing that these distros easily exceed 7 Gigs of space. Simply selects the bootup drive as the USB, boot the PC and follow the prompts. There are many documents which describe the setup of Linux distros for PCs. As an abstracted explanation, the responsible person must then install Moodle on such a PC, which may entail LAMP or XAMP as pre-requisite. The PC must then be incorporated on the school's network and made available to everyone. To obtain support from all teachers, Moodle may be made available for all subjects, which may necessitate some interest group, convening once a week to discuss Moodle and its rollout.

If the school has finances, just register a domain or use the school's domain and add-on memory storage to the domain at minimal cost. Install Moodle in the cloud through cpanel by verifyin the school's service provider possibilities. Please note that open source and proprietary driven domains will influence these decisions.

7.5.2.5 Learning

A bottom-up approach should be followed before the proposed conceptual framework is implemented. A high level of cohesion should exist between mathematics and programming subjects and the responsible teachers. Metacognition should be practiced, which implies a teacher with passion and integrity that will promote the programming language belief system of learners. Learning can only take place if the teacher implements the proposed conceptual framework with a passion for programming and mathematics for learning to take place.

7.5.2.6 APOS

Mathematics and programming teachers must understand APOS and what is done in programming should complement mathematics teachings by integrating mathematical concept definitions into APOS theory. Mathematical teachers must assist learners in understanding mathematics concept definitions through the programming language analogy.

7.5.2.7 Education

The educational landscape in SA propagates high road transfer of knowledge among learners, through the CAPS curricula at school and tertiary curricula in higher education. These curricula are designed according to best practices around high road transfer of learning. Lee and Choi (2017) see high order thinking as a critical predictor of success. However, students struggle to deliver upon entering industry once finished studying. According to Wilhelm (2008), students may know terminologies, but making connections in applying their knowledge show low road transfer of learning. Low road transfer of learning refers to memorisation and rote learning. Wilhelm (2008) advocates high road transfer of learning, such as this research provides to ensure abstraction through mental processes,

making connections from what were studied at school to new problems confronted with in industry. This research ensures a positive transfer of learning, as APOS theory as lens used within a programming language improves computational thinking, which should have a positive transfer of learning effect on mathematical problem solving. The actual teaching of mathematics may cause a negative transfer of learning, as it necessitates a low road transfer of learning in the form of rote learning concept images through memorisation, as pointed out throughout this research. Educators should employ the proposed conceptual framework to ensure high road transfer of learning.

CHAPTER 8: CONTRIBUTION, FURTHER RESEARCH AND REFLECTIONS

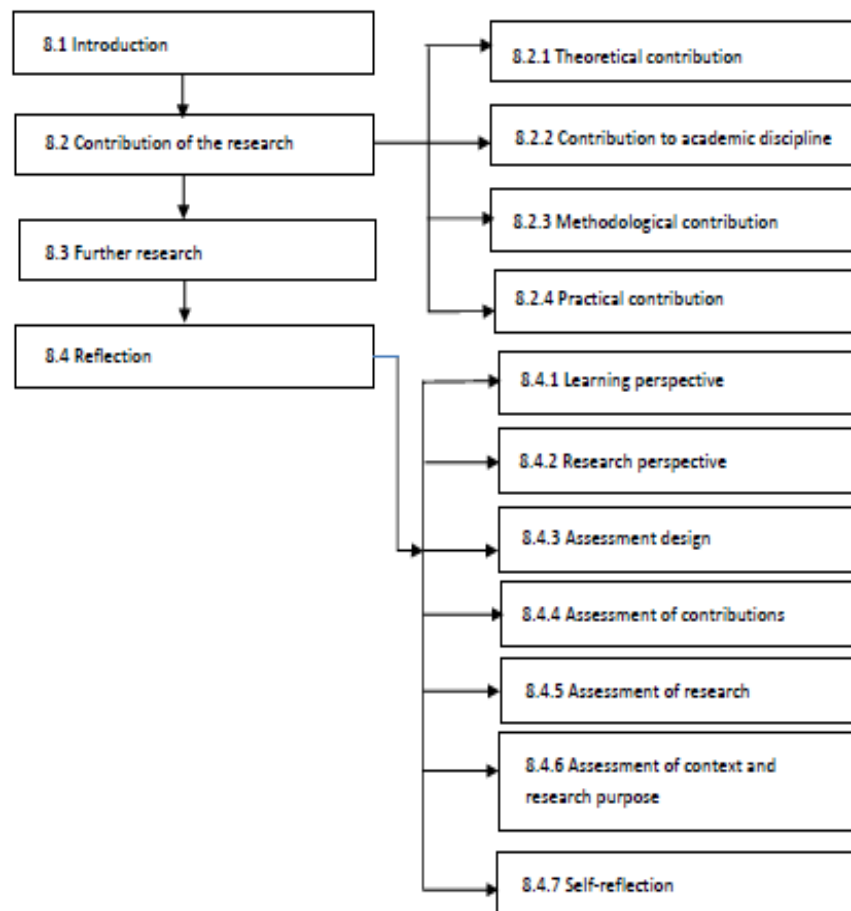


Figure 8.1: Chapter 8 Layout

8.1 Introduction

This Chapter as depicted in [Figure 8.1](#) is divided into three sections, namely contribution of this research, reflection, and further research based on the recommendations in Chapter 7.

The contributions of this research entail:

- i) Psychology of didactics,
- ii) educational practice,
- iii) curriculum design,
- iv) instructional design,
- v) programming expertise in designing the architecture of the research,
- vi) expertise in rolling out the hardware under different circumstances, i.e. cloud-based or locally installed systems at a school's premises and
- vii) the actual research based on EDR,
- viii) research didactics,

- ix) programming language expertise, and finally,
- x) teaching and learning expertise.

The most encompassing and satisfying contribution made through this research is that the learner may know that education in a programming language is a choice which becomes reality if the educational process followed adheres to cognitive load theory, which encompasses all the themes identified in this research.

8.2 Contributions of the research

Te'eni et al. (2015:564) indicate three success components for research, namely “contribution, contribution and contribution”. The authors are critical of theoretical and empirical contributions and urge researchers to make findings exciting by placing it into context relevant outside the scientific study as well.

Ågerfalk (2014) advocates the combination of practice with research. The mixed methods approach in papers is more important than merely including statistical presentations. The research should tell a story, and according to Baskerville (2009), the reader will be able to engage in a process by applying the theory embedded in proposed IT artefact, which was an outcome of this research.

This research study contributed to the existing body of knowledge. The four types of contributions, namely theoretical, academic discipline knowledge, methodological and practical contributions are discussed in the following sub-sections (Hofstee, 2009; Jansen, 2012).

8.2.1 Theoretical contribution

This research aimed to explore and understand how a programming language, using Action Process Object Schema (APOS) theory as lens, could promote computational thinking skills at a cognitive level of formal operations among high school learners (section 7.2.1). Teachers and researchers may promote computational thinking by following the protocol as developed in this research. An approach and attitude, in contributing to the learner's belief system, towards APOS theory was formed through the development of algorithms and mapped into Greenfoot to solve real life problems. The researcher explored the disconnect that exists between computational thinking and the learners. The researcher further investigated the challenges learners experienced when applying computational thinking using a programming language and how they overcame the challenges through computation. The findings function as a driving force for programming language teaching to promote computational thinking as prescribed by the conceptual framework (Figure 8.4).

8.2.2 Contribution to academic discipline

According to Maree (2012), academic disciplines evolve when participants and the respective context are enriched by research. The learners and researcher gained qualities through conducting this research, including developing new skills that enriching their personal environment as well as the school and broader community.

The learners who participated in, and engaged with the research process and protocol, in partnership with the researcher, contributed to the IT, Mathematics and Science disciplines at the private school. The advancement of the academic discipline cannot be ignored as a positive outcome within the community. Furthermore, the research impacted on the learners' subject choices for grade 10, and positively influences teachers' perception on their expanded knowledge of programming competencies and their approach towards computational thinking.

8.2.3 Methodological contribution

Many methodologies were researched, compared and applied towards developing tools for conducting this research, including the following:

- i) APOS theory,
- ii) Genetic decompositions through ACE,
- iii) Schema development methodologies,
- iv) EDR research methodology, and a
- v) technical, pedagogical, and content framework.

The above methodologies were applied to address a broken belief system about mathematics, accommodate a new programming-belief system, and deliver a proposed conceptual framework to amend these belief systems. The proposed conceptual framework (section 8.4.2, Figure 8.4) was constructed from the initial theoretical conceptual framework (section 2.2.3; Figure 2.21) to facilitate the usage of Greenfoot as a programming language in a controlled constructionist manner (section 7.2.2).

The introduction of a programming-belief system and revisiting a broken belief system about mathematics was accomplished through EDR. The EDR was based on the van Wyk and de Villiers model (2018) and subsequent interventions built on APOS theory in applying this model. The framework showed the programming concepts as identified in the literature study, implemented with Greenfoot, and presented in a way that computational thinking was promoted among the partaking learners through APOS theory. The programming language was initially regarded by learners as a threat, as described by the analogy of psychological status in mathophobia that exists within the mathematics learner (Papert, 1980). However, introducing Greenfoot with its immediate response, embodied cognition, and two-dimensional

user-interface brought some calmness to learners and they realised that mastering Greenfoot was easier than they thought it would be. The addition of a lapel badge with the Greenfoot logo (Appendix T) presented to the learners also introduced a matter of pride to their participation in the research.

Another aid was the Moodle LMS, which worked well for some but was perceived as a burden for others who carried a heavy extra-mural load of activities. The LMS provided a cognitive balance to the partaking learners and alleviated many of the unnecessary stress and uncertainty in what to do and how to approach certain constructs using Greenfoot. The major important aspect here was to involve the headmaster in the research. The LMS provided a way for learners to complete their tasks faster.

8.2.4 Practical contribution

This research produced a guideline for radical change and regulation in provisioning the rollout of a new curriculum in parallel with the existing curriculum on the Greenfoot programming language and APOS theory. The abstract part (radical change), according to Cronje (2011), that explores the programming language concepts to determine which programming language concepts are of importance to assist with computational thinking, were addressed (section 2.2.2.1(b)). The target audiences (teachers and researchers) can now follow protocol to build an environment from the ground up to promote computational thinking among learners through replicating this research. This may be done by asking questions such as “What programming language concepts are available to facilitate computational thinking?” and “How can these programming language concepts be implemented in a language like Greenfoot to promote learners’ computational thinking skills within the curriculum?”

The concrete part of the practical contribution, according to Cronje (2011), is the rationale of the exploration which coincides with the EDR strategy used in this research. The Moodle website was structured through a storyboard technique by the researcher in his capacity as a qualified and experienced instructional designer. This configuration is depicted in a final proposed conceptual framework as outcome (Figure 8.4), functioning as a guide for others to promote computational thinking using the Greenfoot programming language. Together with the framework, the LMS, the scenarios within the Greenfoot language created and complimented by the flipped classroom techniques informed the artefact as an enabler for computational thinking. All the exercises on the Moodle website may be used as practical examples by other researchers, academia or educationists in high schools when replicating this research. An instructional designer should develop Moodle content to promote didactics.

8.3 Further Research

Although three categories of teachers (Appendix A-8.1.1 to Appendix A-8.3.5) evaluated the adoption of the innovation, further research needs to be done on the adoption of the innovation when conducting local and broad impact evaluations as discussed by Bannan (2013). Further research requires a positivistic approach within a developmentalist paradigm to be adopted (Weber, 2010) (section 3.2.3.2, Figure 3.4). It is still unclear what role the mathematics teacher plays in understanding the APOS theory, programming language and LMSs because of APOS being unfamiliar to most mathematics teachers in SA. However, mathematics and programming teachers should maintain a high level of cohesion to ensure APOS theory is applied in both domains.

Barrouillet (2015) also distinguishes actions from gestures, which highlighted a concern in Piaget's theories. Gestures can be investigated in future research, especially when focusing on teaching and learning among deaf and blind learners who may use programming to promote their computational thinking.

This study focused on one private school only. It is therefore recommended that more schools, including public schools, are approached. Preferably, a larger number of learners commencing with grade 8 should participate using a programming language and APOS theory as lens. The power of Greenfoot is found in it being a dual-modality (Szabo et al., 2019), visual programming language, but it also provides a progression path into syntax-driven coding. The learners are embodied in the actual language as computational notation instead of merely remaining at a visual level, compared to other languages used in education.

The positive impact of this research on teachers and learners as a whole opens so many avenues for further research in education, even when only the few aspects as mentioned above are considered.

8.4 Reflection

8.4.1 Learning perspective

Reflection from a learning perspective enables the researcher to dynamically change his approach; this approach is also known as Kolb's experiential learning cycle (Kolb, 1984). An example from this research is where the students were asked to state the sum of the angles of a triangle, which resulted in many incorrect answers (section 5.2, Figure 5.3). Through reflection, the researcher envisaged Figure 2.13 as a possible approach for learners to solve the problem of knowing the sum of the angles of a triangle. Such teaching and learning activities through reflection were followed in this research as prescribed by Andresen, Boud

and Cohen (2000). The researcher intervened by influencing the learner's thoughts to take action, by tearing the corners of the drawn triangle and placing them on a ruler to form a straight line, as depicted in Figure 2.13. The activity can be grouped under the Action phase of APOS theory. The authors further identify reflection as a key element of learning from experience, used during this research.

8.4.2 Research perspective

This research, driven by interventions, compelled the researcher to improve on these interventions by develop artefacts in the form of practical outcomes and a theoretical outcome.

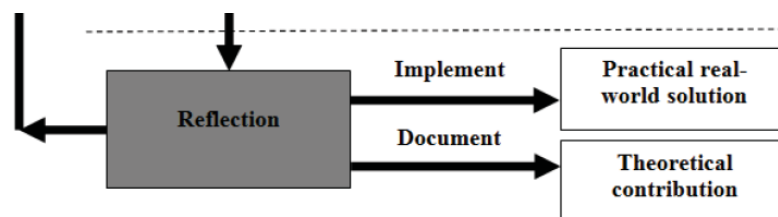


Figure 8.2: Reflection of this EDR research (Adopted from van Wyk & De Villiers, 2018:305)

Reflection is therefore not a separate task; it forms an integral part of the van Wyk and de Villiers (2018) EDR model as well as of this research (Figure 8.2). Upon reflection, this research produced real-world solutions in the form of the Greenfoot programming language APOS approach, structured content within a Moodle LMS, which can be transferred to any LMS, as well as the theoretical component, which is a conceptual framework for teachers and researchers to follow when this research is replicated. Figure 8.3 illustrates the flow of this research's processes. For further research, the instantiation of the artefact may take the form of a construct, model, method, instantiation or combinations of the former (Weber, 2010; Vaishnavi & Kuechler, 2008); Gregor & Hevner, 2013). Weber (2010) further argues that the socio-technologist or developmentalist paradigm forms part of EDR paradigm, hence the double arrows even at the final artefact's construction. The theoretical contribution informs the proposed conceptual framework (Figure 8.4) as theoretical artefact.

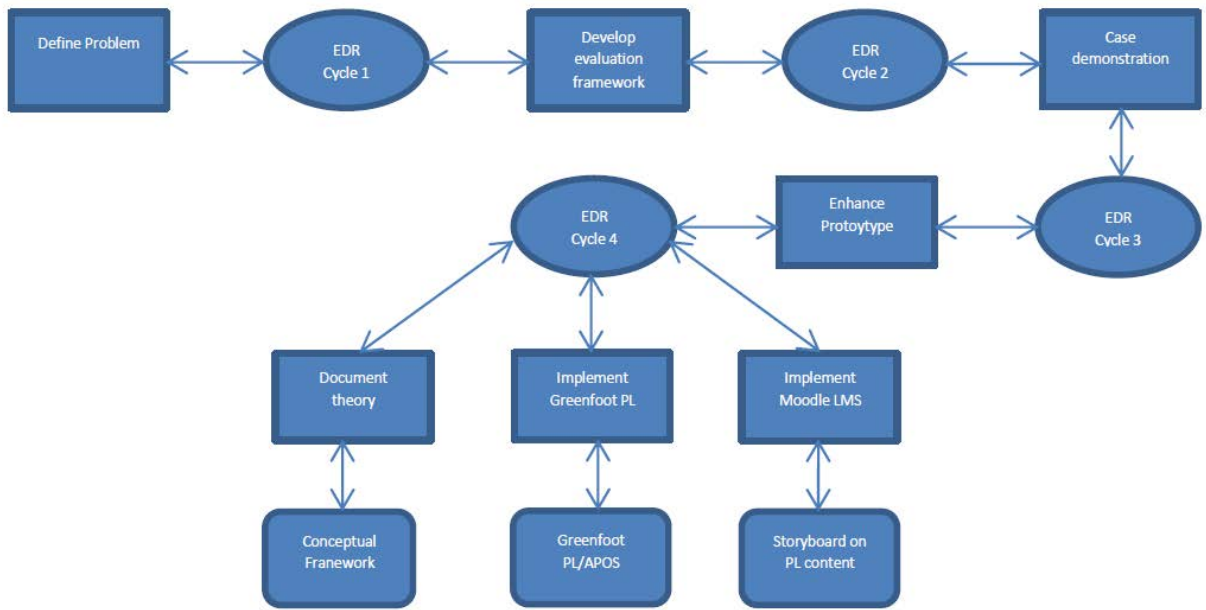


Figure 8.3: Processes flow diagram of this research (Adapted from Van Wyk & De Villiers, 2018:306)

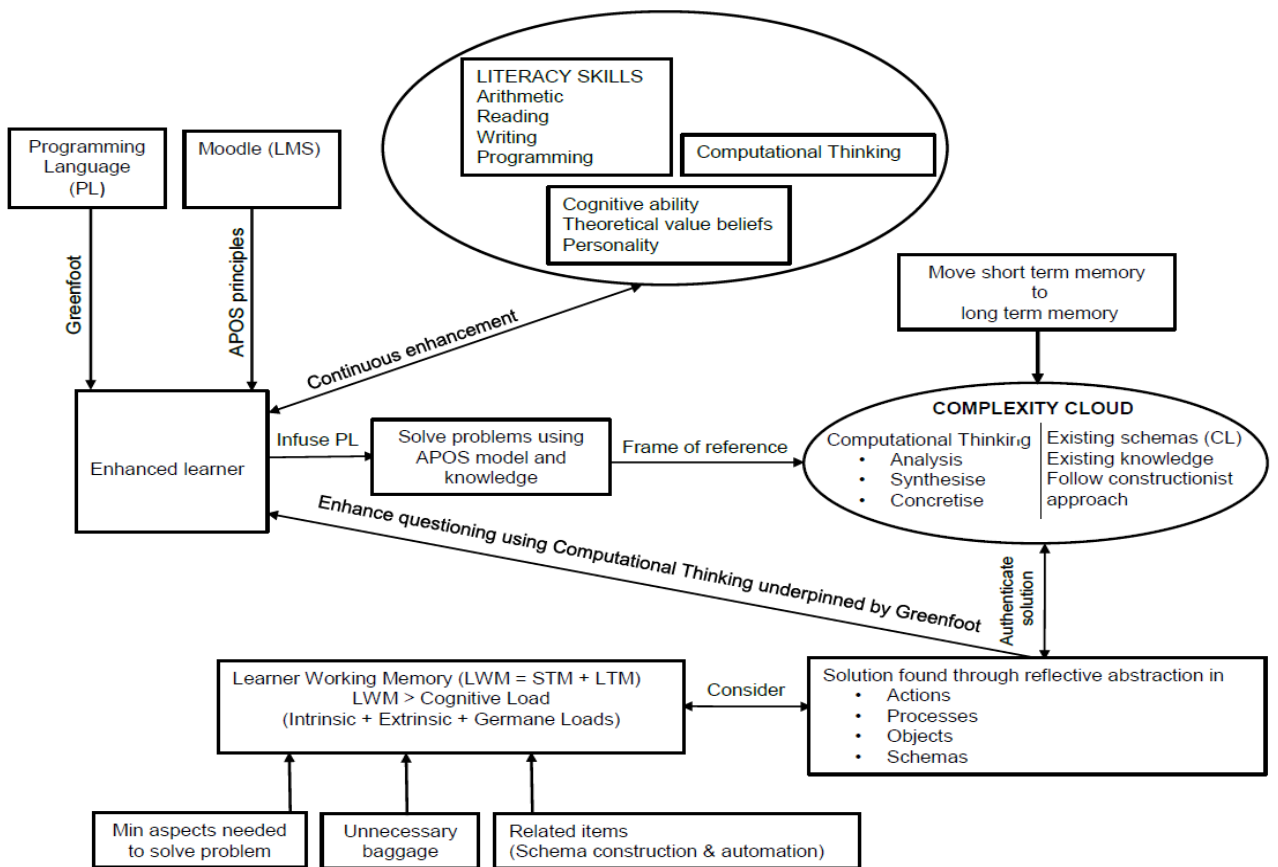


Figure 8.4: Proposed conceptual framework

8.5 Assessment of research

This research is built on a wicked problem, as stated in the educational research problem. South Africa is in dire straits with mathematics teaching and learning. Coding is becoming a popular topic/subject at schools and more people begin to understand what coding entails. Both the conceptual framework and the LMS are artefacts providing a guideline to teachers to replicate this research. The aim of this research was an exploration of how a programming language and APOS theory could promote computational thinking at a cognitive level of formal operations. This research furthermore provided a theoretical component to substantiate the artefacts that were created.

8.6 Assessment of the context and research purpose

The researcher's choice of a private school to conduct the research provided a stable environment for the research. The private school offered Java as programming language and it was easy for the researcher to add Greenfoot to the mix. The terminal setup for the technical platform created challenges in terms of slow compilation speed. However, the EDR processes with abductive inquiry contributed to the success of the results. EDR added structure, technique, reliability, rigour, correctness and validity to the artefacts.

8.7 Self-reflection

Programming formed part of my life since 1981, when I started with a first course in data processing based on card reader technology. I qualified as a teacher with a diploma in education and a second higher teacher's diploma with Mathematics and Biology as majors at Stellenbosch University. The first computer I purchased was the BBC Model B micro-computer, and I taught myself the art of programming in the BASIC programming language by watching many videos made by BBC similar to the flipped classroom technique in 1983. The difference was that these flipped classrooms ran on BETA Max tape technology. Research done on the BBC Model B micro-computer gave birth to the release of the BBC Master computer, which had better peripheral devices. I used to develop a library program/database in BASIC for schools at the time. At the school I taught, I was responsible for the procurement of computers and wrote manuals on word processing and held several teacher training sessions. At the time, I held fund raisers at the school to motivate the purchasing of 7 BBC model B micro-computers. I joined a committee that consisted of 4 members who arranged talks at school centres in the Western Cape. I also integrated the BBC Model B micro-computer into my classroom setting, as I was responsible for gifted child education and could make rapid progress with teaching programming to these gifted children. This involved LOGO as a programming tool and I co-written manuals on LOGO to help pupils gain access to programming. I soon started importing ROM chips from Britain to enhance these BBC micro-computers in offering WordWise and spreadsheet computing for

teachers in 1983. Today I still own three BBC model B micro-computers. This led to an interest in building and repairing computers, which encouraged a computer business in 1989.

In 1985, I furthered my studies to obtain a BSc degree in Computer Science at Stellenbosch University. The passion to educate pupils and students in the art of programming, database management systems, and IT in general became a way of life.

I became involved in industry as DBA-analyst to further my knowledge, where I rolled out Microsoft and Linux servers to provide IT solutions for SME/SMMEs. My passion was still in mathematics and computer programming. I could see the advantage of programming in computational thinking, but never had the time to write up my multiple experiences with gifted learners.

I became an instructor in Oracle products such as Java, Alice and Greenfoot. I noticed the potential of Greenfoot and decided to register for a doctorate qualification to conduct research in my field of passion. I also became a BlackBoard storyboard developer or instructional designer (ID) and rolled out several projects in storyboarding programming and educational modules. This taught me how to structure content on an LMS, governed by moderators, to promote learning. During my research journey, I discovered APOS theory and met up with Prof Dubinsky, the founder of APOS theory, who studied under Piaget. After many discussions with officials of the Western Cape Education Department (WCED), I was granted opportunities throughout the Western Cape at different school centres to address teachers and train them in Greenfoot using APOS theory. Oracle had an agreement with the WCED at the time because of the Java programming language being taught in schools in the Western Cape. I had the opportunity to talk to heads of departments of mathematics and computing at several schools and soon realised that APOS theory may affect our education in a positive way. This gave me a sense of the challenges learners and teachers experienced in our educational system on any subject that necessitates computational thinking.

After the DBE phased out Java PL at schools and opted for a proprietary programming language, I joined a private school and executed my research at a private school in Durbanville in 2014/15. In 2014 I entered into a competition on innovative teaching and learning around mathematics and programming at a conference in Hatfield, England, and managed to be among the top ten innovators where I pitched the idea to the international academic community at Hatfield University in England. This also led to the publication of a paper on mathematics and programming at the time.

The thesis provides me with an opportunity to share my IT knowledge with fellow colleagues and learners on all facets that relate education to programming and to the environment to

replicate the research. From a technical perspective, the research provides a protocol that forms part of prescribed system architecture to structures an educational system, using an inexpensive hardware platform. The complex nature of the 4IR challenge faced by developing countries in Africa may be addressed without venturing into expensive robotics equipment to educate learners in computational thinking. I noticed with a sense of excitement the learners who participated in this research realised that education in mathematics and programming is about making a choice to become part of the 4IR challenge. It is my wish that someone reading this thesis will embrace the ideas and make a difference in the lives of those learners who desperately seek a better life through education.

In conclusion, I side with Baxter, Dubinsky and Levin (1989:v) that “learning” is superior to “teaching” in that what the learner or student does; in other words, to learn is superior to what the educator does to teach.

REFERENCES

- CDE see Centre for Development and Enterprise.
- DBE see Department of Basic Education, South Africa.
- HSRC see Human Sciences Research Council.
- Abdul-Rahman, S.S. & Du Boulay, B. (2014). Learning programming via worked-examples: Relation of learning styles to cognitive load. *Computers in Human Behavior*, 30:286–298.
- Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of Learning Group Publication*, 5(3):438.
- Ågerfalk, P.J. (2014). Insufficient theoretical contribution. A conclusive rationale for rejection? *European Journal of Information Systems*, 23(6):593–599.
- Aho, A.V. (2011). Ubiquity Symposium: Computation and computational thinking. *Ubiquity*, Article 1:1-8. 1–2. ACM. doi:10.1145/1895419.1922682.
- Aho, A.V. (2012). Computation and computational thinking. *The Computer Journal*, 55:7.
- Allison, P. & Pomeroy, E. (2000). Epistemology concerns in research in experiential education. How Shall we “know?” *Journal of Experiential Education*, 23(2):91.
- Alomari, M., El-Kanj, H., Alshdaifat, N. & Topal, A. (2020). A framework for the impact of human factors on the effectiveness of learning management systems. *IEEE Access*, 8, 23542–23558. <https://doi.org/10.1109/ACCESS.2020.2970278>.
- Andrade, M. & Bunker, E.L. (2010). The role of SRL and TELEs in distance education: Narrowing the gap. In *Fostering self-regulated learning through ICT*. IGI Global: 105-121. doi:10.4018/978-1-61692-901-5.ch007.
- Andresen, L., Boud, D. & Cohen, R. (2000). Experience-based learning. In Foley, G. (Ed.). *Understanding adult education and training*. Sydney: Allen and Unwin: 225–239.
- Arnon, I., Cotrill, J., Dubinsky, E., Oktac, A., Fuentes, S.R., Trigueros, M. & Weller, K. (2014). *APOS theory: A framework for research and curriculum development in mathematics education*. New York: Springer-Verlag.
- Bachelard, G. (1938, reprinted 1983). La Formation de l'Esprit scientifique, Contribution à une psychanalyse de la connaissance objective. *Revue de Métaphysique et de Morale*, 45(4):5–7.
- Baist, A. & Pamungkas, A.S. (2017). Analysis of student difficulties in computer programming. *Jurnal Ilmiah Pendidikan Teknik Elektro*, 2(2):81–92. October.
- Bannan, B. (2013). The integrative learning design framework: An illustrated example from the domain of instructional technology. In Plomp, T. & Nieveen, N. (Eds.), *An introduction to educational DR. Proceedings*. Seminar conducted at the East China Normal University, Shanghai, China, 23-26 November, 53–72.

- Barker, R.J. & Unger, E.A. (1983). A predictor for success in an introductory programming class based upon abstract reasoning development. *Proceedings*. 14th SIGCSE Technical Symposium on Computer Science Education of the ACM, Orlando, Florida, 17-18 February.
- Barr, D., Harrison, J. & Conery, L. (2011). Computational thinking: A digital age skill for everyone. The National Science Foundation has assembled a group of thought leaders to bring the concepts of computational thinking to the K-12 classroom. *Learning & Leading with Technology*, 38(6):20–23.
- Barr, V. & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1):48–54.
- Barrouillet, P. (2015). Theories of cognitive development: From Piaget to today. *Developmental Review*, 38(C):1–12. <https://doi.org/10.1016/j.dr.2015.07.004>.
- Baskerville, R. (2008). What DS is not. *European Journal of Information Systems*, 17(5):441–443.
- Baskerville, R. (2009). The EJIS editorial organisation and submissions. *European Journal of Information Systems*, 8(1):1–3.
- Baxter, N., Dubinsky, E. & Levin, G. 1989. *Learning discrete mathematics with ISETL*. New York: Springer-Verlag.
- Beck, R. & Weber, S. (2013). Enhancing IT artefact construction with explanatory and predictive knowledge in DS Research. *Journal of Information Technology Case and Application Research*, 15(1): 4–18. <https://doi.org/10.1080/15228053.2013.10845714>.
- Bennedsen, J. & Caspersen, M. (2019). Failure rates in introductory programming: 12 years later. *Association of Computing Machinery Inroads*, 10(2):30–36. <https://doi.org/10.1145/3324888>.
- Biehler, R.F. & Snowman, J. (1986). *Psychology applied to teaching*. Boston: Houghton Mifflin.
- Blocka, K. (2017). *EEG (Electroencephalogram)*. Reviewed by D. Weatherspoon. [Online]. Available: <https://www.healthline.com/health/eeg>. [Accessed: 11 April 2020].
- Bocconi, S., Chiocciariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P. & Punie, Y. (2016). *Developing computational thinking in compulsory education - Implications for policy and practice*. European Commission, JRC Science for Policy Report. Publications Office of the European Union. doi:10.2791/792158 (online)/10.2791/715431 (ePub).
- Boell, S.K., Cecez-Kecmanovic, D. (2014). A hermeneutic approach for conducting literature reviews and literature searches. *Communications of the Association for Information Systems*, 34(12):257-286.
- Bormanaki, H.B. & Khoshhal, Y. (2017). The role of equilibration in Piaget's Theory Of Cognitive Development and its implication for receptive skills: A theoretical study. *Journal of Language Teaching and Research*, 8(5):996–1005.

- Bosch, M., Gascón, J. & Trigueros, M. (2017). Dialogue between theories interpreted as research praxeologies: The case of APOS and the ATD. *Educational Studies in Mathematics*, 95(1):39–52. <https://doi.org/10.1007/s10649-016-9734-3>.
- Bosse, J., Lee, T.D., Swinson, M. & Faulconer, J. (2010). The NCTM process standards and the five E's of science, connecting math and science. *School Science and Mathematics*, 110(5):262–276. https://onlinelibrary.wiley.com/doi/full/10.1111/j.1949-8594.2010.00033.x?casa_token=SHCvLmG.
- Boyle, T., Bradley, C., Chalk, P., Jones, R. & Pickard, P. (2003). Using blended learning to improve student success rates in learning to program. *Journal of Educational Media (Special Edition on Blended Learning)*, 28(2):165–178. <https://doi.org/10.1080/1358165032000153160>.
- Brennan, K. (2012). ScratchEd: Developing support for educators as designers. In Reilly, E. & Literat, I. (Eds.), *Designing with teachers: Participatory professional development in education*. MIT.
- Brennan, K. & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *American Educational Research Association*. Vancouver: MIT.
- Brijlall, D. & Maharaj, A. (2014). Exploring support strategies for high school mathematics. *International Journal of Science Education*, 7(1):99-107.
- Brijlall, D. & Ndlovu, Z. (2013). High school learners mental construction during solving optimisation problems in Calculus: A South African case study. *South African Journal of Education*, 33(2):1–18.
- Brousseau, G. (1983). Les obstacles épistémologiques et les problèmes en mathématiques. *Recherches en didactique des Mathématiques*, 4(2):164–198.
- Brousseau, G. (2002). *Theory of didactical situations in mathematics*. New York: Kluwer Academic.
- Brousseau, G. (2010). *Glossaire de quelques concepts de la théorie des situations didactiques en mathématiques*. http://guy-brousseau.com/wp-content/uploads/2010/09/Glossaire_V5.pdf. [Date of access: 20 August 2014].
- Brousseau, G., Sarrazy, B. & Novotná, J. (2014). Didactic contract in mathematics education. In Lerman, S. (Ed.), *Encyclopedia of Mathematics Education*. Dordrecht: Springer.
- Brownell, W.A. (1935). Psychological considerations in the learning and the teaching of arithmetic. In *Teaching of arithmetic*. New York: Teachers College: pp.1–31.
- Bruner, J.S. (1966). *Towards a theory of instruction*. New York: Norton.
- Budd, K., Carson, E., Garelick, B., Klein, D., Milgram, R.J., Raimi, R.A., Schwartz, M., Stotsky, S., Williams, V. & Wilson, W.S. (2005). *Ten myths about mathematics education and why you shouldn't believe them*. [Online]. Available: <http://www.math.jhu.edu/~wsw/ED/10myths.pdf>. [Accessed: 23 May 2015].
- Bule, E. & Seith, E. (2012). *Time for young scots to switch on to computer programming*. The Times Educational Supplement Scotland. [Accessed: June 2013].

- Burrell, G. & Morgan, G. (1979). *Sociological paradigms and organisational analysis*. London: Heinemann.
- Burrell, G. & Morgan, G. (2016). *Sociological paradigms and organisational analysis. Elements of sociology of corporate life*. London and New York: Routledge; Taylor & Francis Group.
- Burton, L. (2004). *Mathematicians as enquirers: Learning about learning mathematics*. Dordrecht, The Netherlands: Kluwer Academic.
- Camillus, J.C. (2008). *Strategy as a wicked problem*. Harvard Business review, May. <https://hbr.org/2008/05/strategy-as-a-wicked-problem>. [Accessed: 20 May 2019].
- Termeer, C., Termeer, J.A.M., Dewulf, A. & Biesbroek, R. (2019). A critical assessment of the wicked problem concept: Relevance and usefulness for policy science and practice. *Policy and Society*, 38(2):167–179. doi:10.1080/14494035.2019.1617971.
- Cartmill, E.A., Beilock, S. & Goldin-Meadow, S. (2012). A word in the hand, action, gesture and mental representation in humans and non-human primates. *Philosophical Transactions of the Royal Society*, 367:129–143. <http://doi.org/10.1098/rstb.2011.0162>.
- Centre for Development and Enterprise. (2014). *What does research tell us about teachers, teaching and learner performance in mathematics?* Media release. <http://www.cde.org.za/what-does-research-tell-us-about-teachers-teaching-and-learner-performance-in-mathematics-2/>. [Date of access: 31 August 2018].
- Cegielski, C. & Hall, D. (2006). What makes a good programmer? *Communications of the ACM*, 49(10):73-75.
- Cetin, I. & Dubinsky, E. (2017). Reflective abstraction in computational thinking. *Journal of Mathematical Behavior*, 47:70–80. doi:10.1016/j.jmathb.2017.06.004.
- Cherry, K. (2014). Formal operational stage of cognitive development. <http://psychology.about.com/bio/Kendra-cherry-17268.htm>. [Accessed: 11 October 2014].
- Chevallard, Y. (1989). On didactic transposition theory: Some introductory notes. In Steiner, H.G. & Hejny, M. (Eds.). *Proceedings*. International Symposium on Selected Domains of Research and Development in Mathematics Education University of Bielefeld, Germany, and University of Bratislava, Slovakia: 51–62.
- Chevallard, Y. (2005). Steps towards a new epistemology in mathematics education. *Paper presented*. Fourth Congress of the European Society for Research in Mathematics Education (CERME4), Sant Feliu de Guíxols, Spain, 17–21 February.
- Chevallard, Y. (2006). Steps towards a new epistemology in mathematics education. In M. Bosch. M. (Ed.). *Proceedings*. Fourth Conference of the European Society for Research in Mathematics Education Barcelona, Spain: Universitat Ramon Llull Editions: 21–30.
- Chirinda, B. & Barmby, P. (2018). South African grade 9 mathematics teacher views on the teaching of problem solving. *African Journal of Research in Mathematics, Science and Technology Education*, 22(1):114–124.

- Clark, R.E., Kirschner, P.A. & Sweller, J. (2010). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, 41(2):75–86.
- Clark, R.E., Kirschner, P.A. & Sweller, J. (2012). Putting students on the path to learning: The case for fully guided instruction. *American Educator*, 6–11. Spring.
- Clarke, V. & Braun, V. (2013). Teaching thematic analysis: Overcoming challenges and developing strategies for effective learning. *The Psychologist*, 26(2):120–123. ISSN: 0952-8229.
- Coghlan, D. & Brannick, T. (2001). *Doing action research in your own organisation*. London: Sage.
- Cohen, H. 2013. Making from Scratch: A Transdisciplinary Research into the Historical and Social Production of Subjectivity. pp. In Morgan, C. & Malva, F. (Eds.), *Activating the inanimate: Visual vocabularies of performance practise*. https://doi.org/10.1163/9781848881211_020. Leiden: BRILL, 209–218.
- Cole, R., Purao, S., Rossi, M. & Sein, M. (2005). Being proactive: Where action research meets design research. *Proceedings*. International Conference on Information Systems (ICIS) | Association for Information Systems (ICIS 2005), Las Vegas, Nevada, USA, 11–14 December, 27. <http://aisel.aisnet.org/jcis2005/27>.
- Collis, J. & Hussey, R. (2014). *Business research: A practical guide for undergraduate and postgraduate students*. 4th ed. Basingstoke, Hampshire: Palgrave Macmillan.
- Connolly, C., Murphy, E. & Moore, S. (2009). Programming anxiety amongst computing students – A key in the retention debate? *IEEE Transactions on Education*, 52(1):52–56. <https://doi.org/10.1109/TE.2008.917193>.
- Cook, J.L. & Cook, G. (2005). *Child development*. London: Pearson Education: Allyn & Bacon.
- Cooper, S., Dann, W. & Pausch, R. (2003). Teaching objects-first in introductory computer science. *Proceedings*. The 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003), Reno, Nevada, USA, 19–23 February.
- Cooper, S., Pérez, L.C. & Rainey, D. (2010). Education K-12 computational learning. *Communications of the ACM*, 53(11):27–29.
- Cottrill, J., Dubinsky, E., Nichols, D., Schwingendorf, K., Thomas, K. & Vidakovic, D. (1996). Understanding the limit concept: Beginning with a coordinated process schema. *Journal of Mathematical Behavior*, 15:167–192.
- Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24(1):87–114. <https://doi.org/10.1017/S0140525X01003922>.
- Creswell, J.W. (2014). Chapter One: The selection of a research approach. In Creswell, J.W. (Ed.), *Research design: qualitative, quantitative, and mixed methods approaches*. Sage: 3–23.
- Cronje, J.C. (2011). The ABC instant research question generator. *Paper presented*. The 4th Building the Scientific Mind Colloquium, Stellenbosch, South Africa, March.

- Cronje, J.C. (2016). Towards an integration of paradigmatic and pragmatic research in information systems. *The Electronic Journal on Information Systems in Developing Countries (EJISDC)*, (77). <https://dialnet.unirioja.es/servlet/oaiart?codigo=5695296>.
- D'Amore, B. (2008). Epistemology, didactics of mathematics and teaching practices. *Mediterranean Journal for Research in Mathematics Education*, 7(1):1–22.
- Damasio, A.R. (1999). *The feeling of what happens*. New York/San Diego/ London: Harcourt.
- Declue, T. (1996). Object-orientation and the principles of learning theory: A new look at problems and benefits. *ACM SIGCSE Bulletin*, 28:232-236. doi:10.1145/236462.236546.
- Dempster, M. & Hanna, D. (2014). *How to write a systematic literature review for psychology research for dummies*. Wiley.
- Denning, P.J. (2009). The profession of IT beyond computational thinking. *Communications of the ACM*, 52(6):28–30.
- Denning, P.J. (2017). Viewpoint remaining trouble spots with computational thinking. *Communications of the ACM*, 60(6):33–38. June.
- Department of Basic Education, South Africa. (2011). *Curriculum and assessment policy statement (CAPS)*. Pretoria: Government of South Africa.
- Department of basic education, South Africa. (2015). *South African Schools Act, 1996. (Act No. 84 of 1996): Amended national norms and standards for school funding*. Government Notice 17/2015. 16 January. http://www.gov.za/sites/www.gov.za/files/38397_gon17.pdf. [Accessed: 15 November 2015].
- Department of Basic Education, South Africa. (2015b). *DBE and Bright media equip maths literacy teachers*. <https://www.education.gov.za/DBEandBrightMediaequipMathsLiteracyteachers.aspx>. [Accessed: 20 March 2017].
- Deschryver, M.D. & Yadav, A. (2015). Creative and computational thinking in the context of new literacies: Working with teachers to scaffold complex technology-mediated approaches to teaching and learning. *Journal of Technology and Teacher Education*, 23(3):411–431.
- Design-Based Research Collective. (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Researcher*, 32(1):5–8.
- Devlin, K. (2003). Why universities require computer science students to take math. *Communications of the ACM*, 46(9):37–39. September.
- Dienes, Z.P. (1960). *Building up mathematics*. London: Hutchinson.
- Dubinsky, E. (1991). Reflective abstraction in advanced mathematical thinking. In Tall, D. (Ed.), *Advanced mathematical thinking*. Netherlands: Springer: 95–126.
- Dubinsky, E. (2000). Mathematical literacy and abstraction in the 21st century. *School Science and Mathematics*, 100(6):289–297. March. doi:10.1111/j.1949-8594.2000.tb17322.x.

- Dubinsky, E. (2018). *Constructivist as opposed to constructionist*. Email communication between researcher and author.
- Dubinsky, E. & Lewin, P. (1986). Reflective abstraction and mathematics education: The Genetic Decomposition of induction and compactness. *The Journal of Mathematical Behavior*, 5:55–92.
- Dubinsky, E. & McDonald, M. (2001). APOS: A constructivist theory of learning in undergraduate mathematics education research. In Holton, D., Artigue, M., Kirchnergräber, U., Hillel, J., Niss, M. & Schoenfeld, A. (Eds.), *The teaching and learning of mathematics at university level*. New ICMI Study Series, Vol. 7. Dordrecht: Springer: 275–282.
- Du Boulay, B., O’Shea, T. & Monk, J. (1989). The black box inside the glass box: Presenting computing concepts to novices. In Spohrer, C. (Ed.), *Studying the novice programmer*. London: Lawrence Erlbaum Associates, 431–446.
- Duval, R.A. (2006). Cognitive analysis of problems of comprehension in a learning of mathematics. *Educational Study in Mathematics*, 61:103–131.
- Ebrahimi, A., Geranzeli, S. & Shokouhi, T. (2013). Programming for children: “Alice and scratch analysis”. *Proceedings*. 3rd International Conference on Emerging Trends of Computer and Information Technology (ICETCIT), Singapore, 6–7 November.
- Eisenkraft, A. & Eisenkraft, N. (2011). When wrong answers receive top grades. *Journal of College Science Teaching*, 41(2):28–31.
- Eom, S.B. (2014). Understanding eLearners’ satisfaction with Learning Management Systems. *Bulletin of the IEEE Technical Committee on Learning Technology*, 16(2):3–6.
- Felleisen, M., Findler, R.B., Flatt, M., Krishnamurthi, S., Barzilay, E., McCarthy, J. & Tobin-Hochstadt, S. (2018). A programmable programming language. *Communications of the ACM*, 61(3):62-71. doi:10.1145/3127323.
- Fetters, M.D., Curry, L.A. & Creswell, J.W. (2013). Achieving integration in mixed methods designs – principles and practices. *Health Service Resources*, 48(6):2134–2156. <https://doi.org/10.1111/1475-6773.12117>.
- Feurzeig, W. & Papert, S. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments*, 19(5):487–501.
- Feurzeig, W., Papert, S., Bloom, M., Grant, R. & Solomon, C. (1970). Programming-language as a conceptual framework for teaching mathematics. *Newsletter Special Interest Group on Computer Uses in Education (SIGCUE) Outlook*, 4(2):13–17.
- Fischbein, E. (1987). *Intuition in science and mathematics: An educational approach*. Dordrecht, Holland: Kluwer.
- Flavell, J.H. (1976). Metacognitive aspects of problem solving. In Resnick, R.B. (Ed.), *The nature of intelligence*. Hillsdale, NY: Erlbaum.
- Flavell, J.H. (1979). Metacognition and cognitive monitoring. A new area of cognitive-developmental inquiry. *American Psychologist*, 34:906–911.

- Fletcher, S.H. (1984). *Cognitive abilities and computer programming*. Research/Technical Report 143. EDRS(ED259700).
- Freshwater, D. & Cahill, J. (2012). Why write? *Journal of Mixed Methods Research*, 6(3):151–53.
- Ghazi, S.R., Khan, U.A., Shahzada, G. & Ullah, K. (2014). Formal operational stage of Piaget's Cognitive Development Theory. An Implication in Learning Mathematics. *Journal of Educational Research*, 17(2):71–84.
- Gleason, N.W. (2018). *Higher education in the era of the fourth industrial revolution*. 1st ed. (2018). Singapore: Springer Singapore. <https://doi.org/10.1007/978-981-13-0194-0>.
- Gomes, A. & Mendes, A.J. (2007). Learning to program-difficulties and solutions. *Proceedings*. International Conference on Engineering Education (ICEE 2007), Coimbra, Portugal, 3–7 September.
- Goosen, L., Mentz, E. & Nieuwoudt, H. (2007). Choosing the best programming language. *Proceedings*. The 2007 Computer Science and IT Education Conference (SAICSIT '07), Port Elizabeth South Africa, October, 269–282.
- Gordon, H.W. (1988). Specialised cognitive function and school achievement. *Developmental Neuropsychology*, 4(3):239–257.
- Gray, E.M. & Tall, D.O. (1994). Duality, ambiguity, and flexibility: A 'proceptual' view of simple arithmetic. *Journal for Research in Mathematics Education*, 25(2):116–140.
- Greene, J.C., Caracelli, V.J. & Graham, W.F. (1989). Toward a conceptual framework for mixed-method evaluation designs. *Educational Evaluation and Policy Analysis*, 11:255–274. doi:10.2307/1163620.
- Greenfoot. (2014). *Joy of code*. [Online]. Available: <https://www.greenfoot.org/doc/joy-of-code>. [Accessed: 23 May 2014].
- Gregor, S. & Hevner, A.R. (2013). Positioning and presenting DS research for maximum impact. *MIS Quarterly*, 37(2):337–355.
- Gregor, S., Müller, O. & Seidel, S. (2013). Reflection, abstraction and theorising in design and development research. *Proceedings*. European Conference on Information Systems ECIS (2013), Utrecht Netherlands, 6-8 June, 74. http://aisel.aisnet.org/ecis2013_cr/74.
- Griffiths, D.H. (1973). *The study of the cognitive development of science students in introductory level courses*. ERIC [ED096108].
- Grossman, P.L. & Lynn, P. (1990). The making of a teacher: Teacher Knowledge and teacher education. New York: Teachers College Press, Columbia University.
- Guba, E.G. & Lincoln, Y.S. (1994). Competing paradigms in qualitative research. In Denzin, N.K. & Lincoln, Y.S. (Eds.), *Handbook of qualitative research*. London: Sage: 105–117.
- Gummesson, E. (2000). *Qualitative methods in management research*. 2nd ed. Thousand Oaks, CA: Sage.

- Guzdial, M. (2008). Education paving the way for computational thinking. *Communications of the ACM*, 51:25-27. <https://doi.org/10.1145/1378704.1378713>.
- Hartley, M.S. & Treagust, D.F. (2014). *Learning Environments Research*, 17(1):95–111. <https://doi.org/10.1007/s10984-014-9157-y>.
- Hayakawa, S. (1949). *Language in thought and action*. New York: Harcourt Brace Jovanovich.
- Hazzan, O. (1999). Reducing abstraction level when learning abstract algebra concepts. *Educational Studies in Mathematics*, 40:71–90.
- Hazzan, O. (2003). How students attempt to reduce abstraction in the learning of mathematics and in the learning of computer science. *Computer Science Education*, 13(2):95–122.
- Heintz, F. & Manilla, L. (2018). Computational thinking for all, an experience report on scaling up teaching computational thinking to all students in a major city in Sweden. *Association for Computing Machinery (ACM) Inroads*, 9(2):65–71. <https://doi.org/10.1145/3159450.3159586>.
- Hevner, A.R. March, S.T., Park, J. & Ram, S. (2004). DS in information systems research. *MIS Quarterly*, 28:75–105.
- Higgins, H.J. & Wiest, L.R. (2006). Individual Interviews as insight into children's computational thinking. *Australian Primary Mathematics Classroom*, 11(1):25–29.
- Hill, J., Houle, B., Merrit, S. & Stix, A. (2008). *Applying abstraction to master complexity, the comparison of abstraction ability in computer science majors with students in other disciplines*. Leipzig, Germany: s.n.
- Hill, J. & Scott, T. (2004). A consideration of the roles of business intelligence and e-business in management and marketing decision making knowledge-based and high-tech start-ups. *Qualitative Market Research: An International Journal*, 7(1):48–57.
- Hofstee, E. (2009). *Constructing a good dissertation: A practical guide to finishing a master's, MBA or PhD on schedule*. Sandton: EPE.
- Howie, S.J. (2004). A national assessment in mathematics within an international comparative assessment. *Perspectives in Education*, 22(2):149–162. June.
- Howie S.J. (2013). Language and other background factors affecting secondary pupil's performance in Mathematics in South Africa. *African Journal of Research in Mathematics, Science and Technology Education*, 7(1):1–20. <https://doi.org/10.1080/10288457.2003.10740545>.
- Hromkovič, J. (2006). Contributing to general education by teaching informatics. In Mittermeir, R.T. (Ed.), *ISSEP 2006*. LNCS, 4226, 25–37.
- Hudak, M.A. & Anderson, D.E. (1990). Formal operations and learning style predict success in statistics and computer science courses. *Teaching of Psychology*, 17(4):231–234.
- Hult, M. & Lennung, S-A. (1980). Towards a definition of action research, a note and bibliography. *Journal of Management Studies*, 17(2):241–250. May.

- Human Sciences Research Council. (2014). *Report on the annual national assessments of 2014. Grades 1 to 6 & 9*. <http://www.hsrc.ac.za/en/news/view/ana-2014>. [Accessed: 13 August 2018].
- Ilfenthaler, D., Masduki, I. & Seel, N. (2011). The mystery of cognitive structure and how we can detect it, tracking the development of cognitive structures over time. *Instructional Science*, 39(1):41–61. <https://doi.org/10.1007/s11251-009-9097-6>.
- Iivari, J. (2007). A paradigmatic analysis of information systems as a DS. *Scandinavian Journal of Information System*, 19(2): Article 5. <http://aisel.aisnet.org/sjis/vol19/iss2/5>.
- Jabareen, Y. (2009). Building a conceptual framework, philosophy, definitions and procedure. *International Journal of Qualitative Methods*, 8(4):49–62. <https://doaj.org/article/90c4fedb2fa541e49be6d0717f28c2de>.
- Jackson, N. (2004). Developing the concept of metalearning. *Innovations in Education and Teaching International*, 41(4):391-403. doi:10.1080/1470329042000276995.
- Jankvist, U. & Niss, M. (2018). Counteracting destructive student misconceptions of mathematics. *Education Sciences*, 8(2):53. <https://doi.org/10.3390/educsci8020053>.
- Jansen, J.D. (2012). The quality of doctoral education in South Africa: A question of significance. In Maree, K. (Ed). *Complete your thesis or dissertation successfully: practical guidelines*. Landsdowne: Juta, 1-11.
- Järvinen, P. (2007). Action research is similar to DS. *Quality and Quantity*, 41(1):37-54.
- Jonker, J. & Pennink, B. (2010). *The essence of research methodology. A concise guide for Master and PhD Students in Management Science*. Heidelberg: Springer.
- Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N. & Grise, K. (2018). Computational what? Relating computational thinking to teaching. *TechTrends*, 62(6):574–584. <https://doi.org/10.1007/s11528-018-0290-9>.
- Keegan, R. (2016). Action research as an agent for enhancing teaching and learning in physical education: A physical education teacher's perspective. *The Physical Educator*, 73(2):255–284. <https://doi.org/10.18666/TPE-2016-V73-I2-6236>.
- Kim, T. & Donaldson, T. (2018). Rethinking right. Moral epistemology in management research. *Journal of Business Ethics*, 148(1):5–20. <https://doi.org/10.1007/s10551-015-3009-2>.
- Koehler, M.J. & Mishra, P. (2009). What is technological pedagogical content knowledge. *Contemporary Issues in Technology and Teacher Education*, 9(1):60–70.
- Kolb, D. (1984). *Experiential learning: Experience as the source of learning and development*. New Jersey: Prentice-Hall.
- Kölling, M. (1999a). The problem of teaching object-oriented programming, Part II: Environments. *Journal of Object-Oriented Programming*, 11(9): 6–12.
- Kölling, M. (1999b). The Blue Language. *Journal of Object-Oriented Programming*, 12:10–17.

- Kölling, M. & Henriksen, P. (2005). Game programming in introductory courses with direct state manipulation. *Proceedings*. The 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2005), Monte de Caparica, Portugal. 27–29 June.
- Kölling, M. (2010a). *Introduction to programming with Greenfoot, object-oriented programming in Java with games and simulations*. New York: Prentice Hall.
- Kölling, M. (2010b). The Greenfoot programming environment. *Association for Computing Machinery Transactions on Computing Education*, 10(4):Article 14. doi:10.1145/1868358.1868361. <http://doi.acm.org/10.1145/1868358.1868361>.
- Kolodner, J.L., Crismond, D., Fasse, B., Gray, J., Holbrook, J., Puntambekar, S. (2003). Putting a student centred learning by Design™ curriculum into practice lessons learned. *Journal of the Learning Sciences*, 12:485–547.
- Koivisto, J.M., Hannula, L., Boje, R.B., Prescott, S., Bland, A., Rekola, L. & Haho, P. (2018). Design-based research in designing the model for educating simulation facilitators. *Nurse Education in Practice*, 29, 206–211. <https://doi.org/10.1016/j.nepr.2018.02.002>.
- Kranch, D. (2010). A study of three instructional sequences for developing computer programming expertise in novice learners. Dissertation presented in partial fulfilment of the requirement for the degree Doctor of Philosophy, Capella University.
- Kramer, J. (2007). Abstraction – The key to computing? *Communications of the ACM*, 50(4):36–42. April.
- Kuechler, B. & Vaishnavi, V. (2011). Promoting relevance in IS research: An informing system for design science research. *Informing Science: The International Journal of an Emerging Transdiscipline*, 14: 124–138.
- Lauffer, A. (2011). Concepts, theories, and classifications. *In Understanding your social agency*. London: Sage: 38–56. doi:10.4135/9781452274690.n3.
- LeDoux, J. (1998). *The emotional brain*. Phoenix: Orion Books.
- Lee, J. & Choi, H. (2017). What affects learner’s higher order thinking in technology-enhanced learning environments? The effects of learner factors. *Computers & Education*, 115:143–152. <https://doi.org/10.1016/j.compedu.2017.06.015>.
- Lee, C-S. & Kolodner, J.L. (2011). Scaffolding students development of creative design skills. A curriculum reference model. *Journal of Educational Technology & Society*, 14(1):3–5. <http://search.proquest.com/docview/2139143858/>.
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1):32–37. doi:10.1145/1929887.1929902.
- Lee, Y-L. (2011). Scratch. *Multimedia Programming Environment for Young Gifted Learners*, 34(2):26–31. <https://doi.org/10.1177/107621751103400208>.
- Leedy, P.D. & Ormrod, J.E. (2014). *Practical research: Planning and design*. Global ed. Pearson Education M.U.A.

- Li, X. (2016). Application of cognitive load theory in programming teaching. *Journal of Higher Education Theory and Practice*, 16(6):57–65.
<http://search.proquest.com/docview/1888963072>.
- Li, X. & Chu, S. (2018). Using design-based research methodology to develop a pedagogy for teaching and learning of Chinese writing with Wiki among Chinese upper primary school students. *Computers & Education*, 126:359–375.
<https://doi.org/10.1016/j.compedu.2018.06.009>.
- Linn, M.C. & Dalbey, J. (1989). Cognitive consequences of programming instruction. In Soloway, E. & Spohrer, J.C. (Eds.), *Studying the novice programmer*. London: Lawrence Erlbaum Associates, 58–62.
- Little, L.F. (1984). The influence of structured programming, gender, cognitive development and engagement on the computer programming achievement and logical thinking skills of secondary students. *Dissertation Abstracts*, A45(6), 1708.
- Livingstone, J.A. (2003). *Metacognition: An overview*. [Online]. Available: <http://www.gse.buffalo.edu/fas/sheell/cep564/Metacog.htm>. [Accessed: 10 January 2014].
- Lockwood, J. & Mooney, A. (2017). *CTen Education: Where does it fit? A systematic literary review*. Maynooth University, National University of Ireland.
- Losh, C.L. (1984). The relationship of student hemisphericity to performance in computer programming courses. *Dissertation Abstracts* A44(7), 2127.
- Louden, K.C. (1993). *PLs, principles and practice*. Boston, MA: PWS Publishing.
- Loughran, J., Milroy, P., Berry, A., Gunstone, R. & Mulhall, P. (2001). Documenting science teachers' pedagogical content knowledge through PaP-eRs. *Research in Science Education*, 31:289–307.
- Maharaj, A. (2013). An APOS analysis of natural science students' understanding of Derivatives. *South African Journal of Education*, 33(1):1–19.
<http://dx.doi.org/10.15700/saje.v33n1a458>.
- Malan, D. & Leitner, H. (2007). Scratch for budding computer scientists. *Proceedings. 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2007)*, Covington, KY, USA, 7–10 March, 39. 10.1145/1227310.1227388.
- Maloney, J.H., Peppler, K., Kafai, Y., Resnick, M. & Rusk, N. (2008). Programming by choice. Urban youth learning programming with scratch. *Proceedings. 39th Technical Symposium on Computer Science Education (SIGCSE'08)*, Portland USA, March. <https://dl.acm.org/doi/proceedings/10.1145/1352135>.
- March, S.T. & Smith, G.F. (1995). Design and natural science research on information technology. *Decision Support Systems*, 15(4):251–266.
- Maree, K. (2012). *Complete your thesis or dissertation successfully: Practical guidelines*. Claremont: Juta.
- Maree, K., Aldous, M., Hattingh, A., Swanepoel, A. & Van der Linde, M. (2006). Predictors of learner performance in mathematics and science according to a large scale study in Mpumalanga. *South African Journal of Education*, 26(2):229–252.

- Marimuthu, M. & Govender, P. (2018). Perceptions of scratch programming among secondary school students in KwaZulu-Natal, South Africa. *The African Journal of Information and Communication (AJIC)*, 21(1):51–80. <https://doi.org/10.23962/10539/26112>.
- Mavilidi, M. & Zhong, L. (2019). Exploring the development and research focus of cognitive load theory, as described by its founders. Interviewing John Sweller, Fred Paas, and Jeroen van Merriënboer. *Educational Psychology Review*, 31(2):499–508. <https://doi.org/10.1007/s10648-019-09463-7>.
- Mazaitis, D. (1993). The object-oriented paradigm in the undergraduate curriculum: A survey of implementations and issues. *SIGCSE Bulletin*, 25(3):58–64.
- McCarthy, J. (1980) Circumscription—A form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2):27–39.
- McGowen, M. & Tall, D. (2010). Metaphor or met-before. The effects of previous experience on practice and theory of learning mathematics. *Journal of Mathematical Behavior*, 29(3):169–179. <https://doi.org/10.1016/j.jmathb.2010.08.002>.
- McKenney, S. & Reeves, T. (2012). *Conducting educational DR: What it is, How we do it, and Why*. London: Routledge.
- McPhail, G. (2016): The fault lines of recontextualisation, the limits of constructivism in education. *British Educational Research Journal*, 42:294–313. April.
- Meerbaum-Salant, O., Armoni, M. & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, 23(3):239–264. <https://doi.org/10.1080/08993408.2013.832022>.
- Merriam-Webster.com. 2019. *Cognitive*. [Online]. Available: <https://www.merriam-webster.com>. [Accessed: 12 October 2019].
- Merriam-Webster.com. 2020. *Promote*. [Online]. Available: <https://www.merriam-webster.com/dictionary/promote>. [Accessed: 7 August 2020].
- Meyer, D. (2010). *A literature review of the product and process of abstraction*. <https://docs.google.com/document/d/1jj1FnxUz6INGajT1hXfuvMZ9sUUmLuJJT58xBqqvec/edit?pli=1>. [Accessed: 14 July 2016].
- McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., Mander, K., & McGettrick, A. (2005). Grand challenges in computing: Education--A summary. *The Computer Journal*, 48(1):42–48. <https://doi.org/10.1093/comjnl/bxh064>.
- Miles, M., Huberman, A.M. & Saldaña, J. (2014). *Qualitative data analysis: A methods sourcebook*. 3rd ed. Thousand Oaks, CA: Sage.
- Miah, S.J., Solomonides, I. & Gammack, J.G. (2019). A design-based research approach for developing data-focused business curricula. *Journal of Education and Information Technologies*. <https://doi.org/10.1007/s10639-019-09981-5>.
- Monfort, M., Martin, S.A. & Frederickson, W. (1990). Information-processing differences and laterality of students from different colleges and disciplines. *Perceptual & Motor Skills*, 70(1):163–172.

- Morine-Dersheimer, G. & Kent, T. (1999). The complex nature and sources of teachers' pedagogical knowledge. In Gess-Newsome, J. & Lederman, N.G. (Eds.), *Examining pedagogical content knowledge*. Dordrecht, The Netherlands: Kluwer Academic, 21–50.
- Moritz, S. & Lysaker, P. (2018). Metacognition – What did James H. Flavell really say and the implications for the conceptualisation and design of metacognitive interventions. *Schizophrenia Research*, 201, 20–26. <https://doi.org/10.1016/j.schres.2018.06.001>.
- Moscucci, M. (2007). About mathematical belief systems awareness. *Proceedings*. CERME 5 Congress of European Society for Research in Mathematical Education Larnaca, Cyprus, 22–26 February.
- Moscucci, M. & Bibbo, C. (2015). About relationships in the affect domain. About relationships in the affect domain conference. *Proceedings*. CERME 9 Ninth Congress of the European Society for Research in Mathematics Education, Prague, Czech Republic, 4–8 February, 1238-1244.<hal-01287351>.
- Moström, J.E. (2011). *A study of students problems in learning to program*. Umea: Department of Computing Science, Umea University, Sweden. ISSN: 0348-0542; ISBN: 978-91-7459-293-1.
- Mostyn, G. (2012). Cognitive load theory: What it is, why it's important for accounting instruction and research. *Issues in Accounting Education*, 27(1):227–245. <https://doi.org/10.2308/iace-50099>.
- Mpofu, R. & Nicolaidis, A. (2019). Frankenstein and the Fourth Industrial Revolution (4IR). Ethics and human rights considerations. *African Journal of Hospitality, Tourism and Leisure*, 8(5):1–25. <https://doaj.org/article/671e14644c3b4e748ad675889e9e0d86>.
- Muis, K.R. (2004). Personal epistemology and mathematics. A Critical review and synthesis of research. *Review of Educational Research*, 74(3):317–377. Fall. ProQuest Educational Journals.
- Nelson, E.E. (2012). The neurobiological basis of empathy and its development in the context of our evolutionary heritage. In Narvaez, D., Panksepp, J., Schore, A.N. & Gleason, T.R. (Eds.), *Evolution, early experience and human development: From Research to practice and policy*. New York: Oxford University Press.
- Nicholls, D. (2009). Qualitative research: Part two - Methodologies. *International Journal of Therapy and Rehabilitation*, 16(11):586–92.
- Nieveen, N. (2013). Educational DR: An introduction. In Plomp, T. & Nieveen, N. (Eds.), *An introduction to educational DR*. Seminar conducted at the East China Normal University, Shanghai, China, 23-26 November: 89–101.
- Norwich, B. & Ylonen, A. (2015). Lesson study practices in the development of secondary teaching of students with moderate learning difficulties: A systematic qualitative analysis in relation to context and outcomes. *British Educational Research Journal*, 41(4):629–649.
- Nunamaker Jr, J.F., Chen, M. & Purdin, T.D.M. (1991). Systems development in information systems research. *Journal of Management Information Systems*, 7(3):89–106.

- Osherson, D., Perani, D., Cappa, S., Schnur, T., Grassi, F. & Fazio, F. (1998). Distinct brain loci in deductive versus probabilistic reasoning." *Neuropsychologia*, 36(4):396-376.
- Ojose, B. (2008). Applying Piaget's Theory of Cognitive Development to mathematics Instruction. *The Mathematics Educator*, 18(1):26–30.
- Onwuegbuzie, A. & Collins, M.T. (2007). A typology of mixed methods sampling designs in social science research. *The Qualitative Report*, 12(2):281–316.
- Op't Eynde, P., De Corte, E. & Verschaffel, L. (2002). Framing students' mathematics-related beliefs. In Leder, G.C., Pehkonen, E., Törner, G. (Eds.), *Beliefs: A hidden variable in mathematics education?* Dordrecht, The Netherlands: Kluwer Academic Publishers: 13–37. ISBN: 1-4020-1058-3.
- Organick, E.I., Forsythe, A.I. & Plummer, R.P. (1989). *Programming language structures*. Academic Press. doi:0-12-528260-5.
- Ott, C.F.P. (1989). Predicting achievement in computer science through selected academic, cognitive and demographic variables. Dissertation Abstracts, A49(10), 2988.
- Overmars, M. (2005). Teaching computer science through game design. *IEEE Computer*, 37(4):81–83.
- Pearsall, J. & Trumble, B. (Eds.) (2002). *Oxford Encyclopedic English Dictionary*. 2nd ed. Oxford: Oxford University Press.
- Page, C. & Meyer, D. (2000). *Applied research design for business and management*. Sydney: McGraw-Hill.
- Pallrand, G.J. (1979). The transition to formal thought. *Journal of Research in Science Teaching*, 16:445–451.
- Papadakis, S., Kalogiannakis, M., Orfanakis, V. & Zaranis, N. (2014). Novice programming environments. Scratch & App Inventor: A first comparison. *Proceedings. 2014 Workshop on Interaction Design in Educational Environments June (IDEE '14)*, New York, USA, 27-30 April, 1–7. <https://doi.org/10.1145/2643604.2643613>.
- Papavlasopoulou, S., Giannkos, M.N. & Jaccheri, L. (2019). Exploring children's learning experience in constructionism-based coding activities through design-based research. *Computers in Human Behavior*, 99, 415–427. <https://doi.org/10.1016/j.chb.2019.01.008>.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.
- Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. NY: Basic Books.
- Papert, S. (2005). Teaching children thinking. *Contemporary Issues in Technology and Teacher Education*, (3):353–365.
- Peabody, J. (2014). *It hurts, but sometimes rote learning is right*. The Times Educational Supplement, London. <https://www.tes.com/news/it-hurts-sometimes-rote-learning-right-0>. [Accessed: 21 July 2020].

- Pearson, K. (2008). From a usable past to collaborative future: African American culture in the age of computational thinking. *Black History Bulletin*, 72(1):41–44.
- Peppers, K., Tuunanen, T., Rothenberger, M. & Chatterjee, S. (2008). A DS research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77.
- Pepin, B. (2014). Using the construct of the didactic contract to understand student transition into university mathematics education. *Policy Futures in Education*, 12(5):646-657. www.worlds.co.uk/PFIE.
- Perkins, D.N. & Salomon, G. (1992). Transfer of learning. In *International Encyclopedia of Education*. 2nd ed. Oxford, England: Pergamon Press. <http://learnweb.harvard.edu/alps/thinking/docs/traencyn.htm>. [Accessed: 31 March 2011].
- Perrenet, C. (2010). Levels of thinking in computer science: Development in bachelor students' conceptualisation of algorithm. *Education and Information Technologies*, 15(2):87–107. doi:10.1007/s10639-009-9098-8.
- Peters, B.G. (2017). What is so wicked about wicked problems? A conceptual analysis and a research program. *Policy and Society*, 36(3):385–396. doi:10.1080/14494035.2017.1361633.
- Philbin, C.A., Bagge, P., Darbyshire, C. & Savage, M. (2013). *Exploring computational thinking*. [Online]. Available: <http://www.google.com/edu/computational-thinking/what-is-ct.html>. [Accessed: 14 July 2013].
- Piaget, J. (1964). Cognitive development in children: Piaget development and learning. *Journal of Research in Science Teaching, Part I*, 2(3):176–186.
- Piaget, J. (1965). *The child's conception of number*. Gattegno, C. & Hodgson, F.M. (trans). New York: W.W. Norton. (Original work published 1941).
- Piaget, J. (1973). Comments on mathematical education. In Howson, A.G. (ed.), *Developments in mathematical education. Proceedings*. Second International Congress on Mathematical Education, Cambridge, UK, 79–87.
- Piaget, J. (1975). *The equilibration of cognitive structures*. Cambridge: Harvard University Press.
- Piaget, J. (1977). *The essential Piaget*. In Gruber, H.E., Voneche, J.J. (eds.). New York.
- Piaget, J. (1985). *The equilibration of cognitive structures*. Cambridge MA: Harvard.
- Piaget, J. & Garcia, R. (1989). *Psychogenesis and the history of science*. Feider, H. (Trans.). New York: Columbia University Press. (Original work published, 1983).
- Plerou, A. (2014). Dealing with Dyscalculia over time. *Proceedings*. Fourteenth Annual International Conference on Information and Communications Technologies in Education, Kos, Greece, 3-5 July, 2014. <https://doi.org/10.13140/2.1.4229.5681>.
- Plomp, T. (2013). Educational DR: An introduction. In Plomp, T. & Nieveen, N. (eds.), *An introduction to educational DR. Proceedings*. Seminar conducted at the East China Normal University, Shanghai, China, 23-26 November, 9–35.

- Portnoff, S. (2018). The introductory computer programming course is first and foremost a language course. *ACM Inroads*, 9(2):34–52. <https://doi.org/10.1145/3152433>.
- Postelnicu, V. (2017). Didactic transposition in school algebra: The case of writing equations of parallel and perpendicular lines. *Proceedings*. Tenth Congress of the European Society for Research in Mathematics Education (CERME 10), Dublin, Ireland, Feb. hal-01914664.
- Prensky, M. (2008). *Programming is the new literacy*. www.edutopia.org/programming-the-new-literacy. [Accessed: 15 June 2014].
- Pries-Heje, J., Baskerville, R. & Venable, J.R. (2008). Strategies for DS research evaluation. *Proceedings*. European Conference on Information Systems ECIS (2008), Galway, Ireland, 9-11 June, 87. <http://aisel.aisnet.org/ecis2008/87>.
- Rahmat, M., Shahrani, S., Latih, R., Yatim, N.F.M., Zainal, N.F.A. & Ab Rahman, R. (2012). Major problems in basic programming that influence student performance. *Procedia-Social and Behavioral Sciences*, 59:287-296.
- Rapoport, R.N. (1970). Three dilemmas in action research. *Human Relations*, (23):499–513.
- Reddy, V., Van der Berg, S., Janse van Rensburg, D. & Taylor, S. (2012). Educational outcomes: Pathways and performance in South African high schools. *South African Journal of Science*, 108(3-4): [online]. ISSN: 1996–7489.
- Reddy, V., Zuze, T.L., Visser, M., Winnaar, L., Juan, A., Prinsloo, C.H., Arends, F. & Rogers, S. (2015). *Beyond benchmarks: What twenty years of TIMSS data tell us about South African education*. HSRC.
- Reeves, T.C. (2006). DR from a technology perspective. In Van den Akker, J., Gravemeijer, K., McKenney, S. & Nieveen, N. (Eds), *Educational DR*. London: Routledge: 52–66.
- Resnick, M. & Ocko, S. (1990). *LEGO/Logo: Learning though and about design*. Epistemology and Learning Group, E&L Memo No. 8, MIT Media Laboratory, Cambridge.
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, November.
- Rittel, H.W.J. & Webber, M.M. (1973). Dilemmas in the general theory of planning. *Policy Sciences*, 4:155–169.
- Rizzolatti, G. & Craighero, L. (2004). The mirror-neuron system. *Annual Review of Neuroscience*, 27:169–192.
- Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172. doi:10.1076/csed.13.2.137.14200.
- Rotenberg, V.S. & Arshavsky, V.V. (1997). Right and left brain hemispheres activation in the representatives of two different cultures. *Homeostasis in Health & Disease*, 38(2):49–57.

- Roth, W.M. & Thom, J.S. (2009). Bodily experience and mathematical conceptions: from classical views to a phenomenological reconceptualisation. *Educational Studies in Mathematics*, 70:175–189. <https://doi.org/10.1007/s10649-008-9138-0>.
- Rovegno, I.C. (1992). Learning to teach in a field-based methods course: The development of pedagogical content knowledge. *Teaching and Teacher Education*, 8:69–82.
- Rutherford, D.G. (2011). *A model of assimilation and accommodation in the cognitive & cultural realms*. Dynamical Psychology. http://dynapsyc.org/2011/Rutherford_2011.pdf. [Accessed: 20 March 2015].
- Rutherford, J. & Ahlgren, A. (1990). *Science for all Americans*. New York: Oxford University Press.
- Saeli, M., Perrenet, J., Swaneveld, B. & Jochems, M.G. (2011). Teaching programming in secondary School. A pedagogical content knowledge perspective. *Informatics in Education*, 10(1):73-88.
- Sanders, L.R., Borko, H. & Lockard, J.D. (1993). Secondary science teachers' knowledge base when teaching science courses in and out of their area of certification. *Journal of Research in Science Teaching*, 30:723–736.
- Saunders, M., Lewis, P. & Thornhill, A. (2019). Chapter 4: Understanding research philosophies and approaches. In *Research methods for business students*. 5th ed. London: Pitman, 28-170.
- Sayed, Y., Motala, S. & Hoffman, N. (2017). *Decolonising initial teacher education in South African universities: More than an event*. South African Education Research Association.
- Schäfer, D. (2018). *Systemic tests: Results show steady improvement in WCape*. <https://www.politicsweb.co.za/politics/2018-systemic-tests-results-show-steady-improvement>. [Accessed: 20 April 2018].
- Schober, A. & Keller, L. (2012). Impact factors for learner motivation in blended learning environments. *International Journal of Emerging Technologies in Learning (iJET)*, 7(2012). <https://www.learntechlib.org/p/44977/>.
- Schoenfeld, A. (1989). Explorations of students' mathematical beliefs and behavior. *Journal for Research in Mathematics Education*, 20(4):338–355. <https://doi.org/10.2307/749440>.
- Schwebel, M. (1975). Formal operations in first year college students. *Journal of Psychology*, 91:133-141.
- Scotland, J. (2012). Exploring the philosophical underpinnings of research: Relating ontology and epistemology to the methodology and methods of the scientific, interpretive, and critical research paradigms. *English Language Teaching*, 5(9):9–16. <http://dx.doi.org/10.5539/elt.v5n9p9>.
- Selby, C. & Woollard, J. (2014). *Refining an understanding of computational thinking*. University of Southampton. <https://eprints.soton.ac.uk/372410/>. [Accessed: 4 September 2019].

- Sfard, A. (1991). On the dual nature of mathematical conceptions: Reflections on processes and objects on different sides of the same coin. *Educational Studies in Mathematics*, (22):1–36.
- Sharma, G. 2017. Pros and cons of different sampling techniques. *International Journal of Applied Research*, 3(7):749-752.
- Shulman, L. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher*, 15(2):4–14. <https://doi.org/10.3102/0013189X015002004>.
- Simon, H. (1996). *The sciences of the artificial*. 3rd ed. Cambridge, MA: MIT Press.
- Skemp, R. (1971). *The psychology of learning mathematics*. New York, NY: Penguin.
- Skemp, R. (1976). Relational understanding and instrumental understanding. *Mathematics Teaching in the Middle School*, 12(2):88–95.
- Skemp, R.R. (1979). *Intelligence, learning and action*. London: Wiley.
- Sriraman, B. & Dickman. (2017). Mathematical pathologies as pathways into creativity. *ZDM – International Journal on Mathematics Education*, 49:137–145. <https://doi-org.libproxy.cput.ac.za/10.1007/s11858-016-0822-8>.
- Spaull, N. (2013). *South Africa's education crisis: The quality of education in South Africa (1994-2011)*. Report Commissioned by CDE. October.
- Suizzo, M. (2000). The social-emotional and cultural contexts of cognitive development: Neo-Piagetian Perspectives. *Child Development*, 71(4):846–849.
- Susman, G.I. & Evered, R.D. (1978). An assessment of the scientific merits of action research. *Administrative Science Quarterly*, 23(4):582–603. December.
- Sweller, J. & Paas, F. (2017). Should self-regulated learning be integrated with cognitive load theory? A commentary. *Learning and Instruction*, 51:85–89. <https://doi.org/10.1016/j.learninstruc.2017.05.005>.
- Sweller, J., van Merriënboer, J.J.G. & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review*. <https://doi-org.libproxy.cput.ac.za/10.1007/s10648-019-09465-5>.
- Szabo, C., Sheard, J., Luxton-Reilly, A., Simon, Becker, B.A. & Ott, L. (2019). Fifteen years of introductory programming in schools: A global overview of K–12 Initiatives.. *Proceedings. 19th Koli Calling International Conference on Computing Education Research (Koli Calling '19)*, Koli National Park Finland, 21-24 November, Article No.: 8:1–9. <https://doi.org/10.1145/3364510.3364513>.
- Szlávi, P. & Zsakó, L. (2017). The cognitive toolkit of programming – Algorithmic abstraction, decomposition-superposition. *Acta Didactica Napocensia*, 10(4):33–40.
- Takeda, H., Veerkamp, P., Tomiyama, T. & Yoshikawam, H. (1990). Modeling design processes. *AI Magazine*: 37–48. Winter.
- Tall, D.O. (2003). Using technology to support an embodied approach to learning concepts in mathematics. In Carvalho, L.M. & Guimarães, L.C. (Eds.), *História e Tecnologia no Ensino da Matemática*, 1:1–28.

- Tall, D.O. (2004). Thinking through three worlds of mathematics. *Proceedings. 28th International Conference of the International Group for the Psychology of Mathematics Education (PME-28)*, Bergen, Norway, 14–18 July: 281–288.
- Tall, D.O. (2008). The transition to formal thinking in mathematics. *Mathematics Education Research Journal*, 20(2):5–24. <http://doi.org/10.1007/BF03217474>.
- Techopedia.com. 2017. *What is OOP?* Techopedia. [Online] Available: <https://www.techopedia.com/definition/3235/object-oriented-programming-oop>. [Accessed: 23 May 2019].
- Te'eni, D., Rowe, R., Ågerfalk, P.J. & Lee, J.S. (2015). Publishing and getting published in EJIS: Marshalling contributions for a diversity of genres. *European Journal of Information Systems*, 24:559–568.
- Tramonti, M., Paneva-Marinova, D. & Pavlov, R. (2017). *Math and art convergence for education*. Central Bohemia University, Prague: 851.
- Trigueros, M. (2005). La noción del esquema en la investigación en matemática educativa a nivel superior. *Educación Matemática*, 17(1):5–31.
- Truran, J. (1992). Integers as jelly beans. *Australian Mathematics Teacher*, 48(2):25.
- Vaishnavi, V. & Kuechler, W. (2008). *Design Science research methods and patterns: Innovating Information and Communication Technology*. Boston, MA: Auerbach.
- Vaishnavi, V., Kuechler, W. & Petter, S. (Eds.) (2004/19). *Design Science Research in Information Systems*. January 20, 2004 (created in 2004 and updated until 2015 by Vaishnavi, V. and Kuechler, W.); last updated (by Vaishnavi, V. and Petter, S.), June 30, 2019. [Online]. Available: <http://www.desrist.org/design-research-in-information-systems/>. [Accessed: 24 March 2019].
- Van den Akker, J. (1999). Principles and methods of development research. In Van den Akker, J., Branch, R.M., Gustafson, K., Nieveen, N. & Plomp, T. (Eds.), *Design approaches and tools in education and training*. Boston: Kluwer Academic, 1–14.
- Van den Akker, J. (2003). Curriculum perspectives: An introduction. In Van den Akker, J. Kuiper, W. & Hameyer, U. (Eds.), *Curriculum landscapes and trends*. Dordrecht: Kluwer Academic, 1–10.
- Van Driel, J.H., Verloop, N. & De Vos, W. (1998). Developing science teachers' pedagogical content knowledge. *Journal of Research in Science Teaching*, 35(6):673–695.
- Van Wyk, E.A. & De Villiers, M.R. 2016. Applying educational design research to virtual reality safety training in mines. *Proceedings of the 15th European Conference on Research Methodology for Business and Management Studies (ECRM 2016)*, Kingston University, London, UK, 9–10.
- Van Gog, T., Paas, F. & Van Merriënboer, J.J.G. (2005). Uncovering expertise-related differences in troubleshooting performance: Combining eye movement and concurrent verbal protocol data. *Applied Cognitive Psychology*, 19(2):205–221.
- Venable, J. (2006). A framework for DS research activities. *Proceedings. 2006 Information Resource Management Association Conference (IRMA)*, Washington DC, USA, 21-24 May, 184–187.

- Venable, J. & Baskerville, R. (2012). Eating our own cooking: Toward a More Rigorous DS of Research Methods. *Electronic Journal of Business Research Methods*, 10(2):141–153.
- Venable, J., Pries-Heje, J. & Baskerville, R. (2016). FEDS: A framework for evaluation in DS research. *European Journal of Information Systems*, 25(1):77–89.
<http://www.tandfonline.com/doi/abs/10.1057/ejis.2014.36>.
- Venable, J.R. & Travis, J. (1999). Using a group support system for the distributed application of soft systems methodology. *Proceedings*. 10th Australasian Conference on Information Systems (ACIS), Wellington, New Zealand, 1-3 December, 1105–17.
- Vidakovic, D., Dubinsky, E., Weller, K. (2018). APOS theory: Use of computer programs to foster mental constructions and student's creativity. In Freiman V. & Tassell J. (Eds.), *Creativity and technology in mathematics education*. Mathematics Education in the Digital Era, vol 10. Cham: Springer.
- Venkatesh, V., Brown, S. & Bala, H. (2013). Bridging the qualitative-quantitative divide: Guidelines for conducting mixed methods research in Information Systems. *Management Information Systems Quarterly*, 37(1):21–54.
- Vergnaud, G. (1990). La théorie des champs conceptuels. *Recherches en Didactique des Mathématiques*, 10(23):133–170.
- Vergnaud, G. (2013). Pourquoi la théorie des champs conceptuels? Por qué la teoría de los campos conceptuales? *Infancia y Aprendizaje*, 36(2):131–161.
- Voogt, J., Fisser, P., Good, J., Mishra, P. & Yadav, A. (2015). CT in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4):715–728.
- Vygotsky, L.S. (1978). *Socio-cultural theory. Mind in society*. Cambridge, MA: Harvard University Press.
- Wademan, M.R. (2005). Utilising development research to guide people-capability maturity model adoption considerations. Doctoral dissertation, Syracuse University. Dissertation Abstracts International, 67-01A, 434. UMI: 3205587.
- Wahyuni, D. (2012). The research design maze: understanding paradigms, cases, methods and methodologies. *Journal of Applied Management Accounting Research*, 10(1):69–80. SSRN: <https://ssrn.com/abstract=2103082>.
- Weber, S. (2010). DS Research: Paradigm or approach? *Proceedings*. Americas Conference on Information Systems (AMCIS 2010), Lima, Peru, 12-15 August, 214.
<http://aisel.aisnet.org/amcis2010/214>.
- Weigend, M. (2006). From intuition to programme. In Mittermeir, R.T. (Ed.), *Informatics education – The bridge between using and understanding computers*. ISSEP 2006. Lecture Notes in Computer Science, 4226. Heidelberg: Springer, Berlin.
- Weintrop, D. & Wilensky, U. (2018a). To block or not to block, that is the question: Students' perceptions of block-based programming. *Proceedings*. 14th International Conference on Interaction Design and Children (IDC '15.), Boston, Massachusetts, June, 199–208.
<https://doi.org/10.1145/2771839.2771860>.


- Weintrop, D. & Wilensky, U. (2018b). How block-based, text-based, and hybrid block/text modalities shape novice programming practices. *International Journal of Child-Computer Interaction*, 17:83–92. <https://doi.org/10.1016/j.ijcci.2018.04.005>.
- White, G. (2003). Standardised mathematics scores as a prerequisite for a first programming course. *Mathematics and Computer Education*, 37(1):96–104.
- White, G.L. & Sivitanides, M.P. (2002). A theory of the relationships between cognitive requirements of computer PLs and programmers' cognitive characteristics. *Journal of Information Systems Education*, 13(1):59-66.
- Wilensky, U. (1991). Abstract meditations on the concrete and concrete implications for mathematics education. In Harel, I. & Papert, S. (Eds.), *Constructionism*. Norwood N.J.: Ablex.
- Wilhelm, J.D. (2008). The problem of teaching for transfer: Taking the low road or the high road? *Voices From the Middle Education Database*, 15(4):45–47.
- Wilson, A. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review*, 9(4):625-636.
- Wilson, A.D. & Golonka, S. (2013). Embodied cognition is not what you think it is. *Frontiers in Psychology*, 4(Article 58). doi:10.3389/fpsyg.2013.00058.
- Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35. March.
- Wing, J.M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A*, 366:3717–3725. doi:10.1098/rsta.2008.0118.
- Wing, J. (2011). *Research notebook: Computational thinking - What and why?* http://link.cs.cmu.edu/files/11-399_The_Link_Newsletter-3.pdf. [Accessed: 25 March 2014].
- Winter, R. (2008). DS research in Europe. *European Journal of Information Systems*, 17(5):470–475.
- Woiceshyn, J. & Daellenbach, U. (2018). Evaluating inductive vs. deductive research in management studies. Qualitative research in organisations and management. *An International Journal*, 13(2):183–195. <https://doi.org/10.1108/QROM-06-2017-1538>.
- Wolcott, M.D., Lobczowski, N.G., Lyons, K. & McLaughlin, J.E. (2019). Design-based research: Connecting theory and practice in pharmacy educational intervention research. *Currents in Pharmacy Teaching and Learning*, 11(3):309–318. <https://doi.org/10.1016/j.cptl.2018.12.002>.
- Wolfram MathWorld. (2020). *Pathological*. <https://mathworld.wolfram.com/Pathological.html>. [Accessed: 25 August 2020].
- Woolfolk, A.E., Winne, P.H. & Perry, N.E. (2003). *Educational psychology*. 2nd ed. Needham Height, MA: Pearson Education Canada: Allyn & Bacon.
- Wright, G., Rich, P. & Leatham, K.R. (2012). How programming fits with technology. *Technology and Engineering Teacher*, 71(3):3–9.

- Yadav, A., Hong, H. & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, 60(6):565–568. <https://doi.org/10.1007/s11528-016-0087-7>.
- Young, G. (2012). *A unitary Neo-Piagetian/Neo-Eriksonian model of development: Fundamental assumptions and meta-issues*. Elsevier.
- Yu, J. & Roque, R. 2019. A review of computational toys and kits for young children. *International Journal of Child-Computer Interaction*, 21:17-36. ISSN 2212-8689. <https://doi.org/10.1016/j.ijcci.2019.04.001>. <http://www.sciencedirect.com/science/article/pii/S2212868918300825>.
- Zeitz, C. & Spoehr, K. (1989). Knowledge organisation and the acquisition of procedural expertise. *Applied Cognitive Psychology*, 3(4):313–336. <https://doi.org/10.1002/acp.2350030404>.
- Zsakó, L. & Szlávi, P. (2012). ICT competences: Algorithmic thinking. *Acta Didactica Napocensia*, 5(2):49–58.

APPENDICES

Appendix A: Introductory letters for the collection of research data

Appendix A-1: Curro Private School



29 November 2013

To Whom It May Concern


I, **Dirk van Zyl**, in my capacity as **Executive Head** at **Curro Durbanville**, give consent in principle to allow **Wilhelm Rothman**, a student at the Cape Peninsula University of Technology, to collect data in this company as part of his D Tech (IT) research. The student has explained to me the nature of his research and the nature of the data to be collected.

This letter of consent in no way commits any individual staff member to participate in the research, and it is expected that the student will get explicit consent from any participants. I reserve the right to withdraw this permission at some future time.

In addition, the company's name may or may not be used as indicated below.
(Tick as appropriate)

	Thesis	Conference paper	Journal article	Research poster
Yes	X	X	X	X
No				

Yours sincerely



DIRK VAN ZYL
Executive Head

Appendix A-2: Ethics Clearance from CPUT



P.O. Box 652 • Cape Town 8000 South Africa • Tel: +27 21 469 1012 • Fax +27 21 469 1002
80 Roeland Street, Vredehoek, Cape Town 8001


Office of the Research Ethics Committee	Faculty of Informatics and Design
--	-----------------------------------

At a meeting of the Faculty Research Ethics Committee on 15 November 2013, ethics approval was granted to MR WILHELM ROTHMAN, student number 190030488 for research activities related to the DTech: Information Technology degree at the Faculty of Informatics and Design, Cape Peninsula University of Technology.

Title of dissertation/thesis:	Programming languages, curriculum and computational thinking at a cognitive level of formal operations
-------------------------------	--

Comments

Research activities are restricted to those detailed in the research proposal. The issues of gatekeepers have been satisfactorily addressed.

 Signed: Faculty Research Ethics Committee	27/11/2013 Date
--	--------------------

Appendix A-3: Chester House Private School



I, Justin Harvey, in my capacity as Headmaster at Chesterhouse School give consent in principle to allow Wilhelm Rothman, a student at the Cape Peninsula University of Technology, to collect data in this company as part of his D Tech (IT) research. The student has explained to me the nature of his research and the nature of the data to be collected.

This letter of consent in no way commits any individual staff member to participate in the research, and it is expected that the student will get explicit consent from any participants. I reserve the right to withdraw this permission at some future time.

In addition, the company's name may or may not be used as indicated below. (Tick as appropriate)

	Thesis	Conference paper	Journal article	Research poster
Yes	✓	✓	✓	✓
No				



 Justin Harvey
 Headmaster

25/11/13

 Date

Appendix A-4: Bosmansdam High Public School



HOËRSKOOL BOSMANSDAM HIGH SCHOOL

(021) 558-1070
FAX (021) 558-6380
headmaster@bosmandam.co.za

Adam Tas / Avenue, BOTHASIG 7441
E-mail hoof@bosmansdam.co.za
E-Mail
Website www.bosmansdam.co.za

I, **Mr D.R. Human**, in my capacity as headmaster at **Bosmansdam High School** give consent in principle to allow Mr W.C. Rothman a student at the Cape Peninsula University of Technology, to collect data in this company as part of his D Tech (IT) research. The student has explained to me the nature of his research and the nature of the data to be collected.

This consent, in no way commits any individual staff member to participate in the research, and it is expected that the student will get explicit consent from any participants. I reserve the right to withdraw this permission at some future time.

In addition, the company's name may or may not be used as indicated below. (Tick as appropriate.)

	Thesis	Conference paper	Journal article	Research poster
Yes				
No				

Mr D.R. Human

15 February 2016

Appendix A-5: Letter of Introduction to Schools



Introductory letter for the collection of research data

Mr W.C. Rothman is registered for the D Tech (IT) degree at CPUT (190030488). The thesis is titled **Programming languages, curriculum and computational thinking at a cognitive level of formal operations in education**, and aims to explore, understand and explain the role and usefulness of a programming language as a means to develop computational thinking skills at a cognitive level of formal operations to facilitate learners to do better at mathematics and science.. The supervisor(s) for this research is/are: Dr AC de la Harpe & Prof Johannes Cronje

In order to meet the requirements of the university's Higher Degrees Committee (HDC) the student must get consent to collect data from organisations which they have identified as potential sources of data. In this case the student will use observation, psychometric tests, pre and post-assessments and interviews to gather data. The psychometric tests will be discussed and negotiated if there is a necessity, but no work will be done without prior consultation and agreement from the school.

If you agree to this, you are requested to complete the attached form (an electronic version will be made available to you if you so desire) and print it on your organisation's letterhead.

For further clarification on this matter please contact either the supervisor(s) identified above, or the Faculty Research Ethics Committee secretary (Ms V Naidoo) at 021 469 1012 or naidoovve@cput.ac.za.

Regards

Dr AC de la Harpe

14 November 2015

Appendix A-6: Videos Uploaded on Moodle Site for Learners to Access

VIDEOS ON JOY OF CODE by Michael Kölling

1. The Joy Of Code#1_ Introduction.mp4
2. The Joy Of Code#2_ Installation.mp4
3. The Joy of Code#3_ Classes and Objects.mp4
4. The Joy of Code #4_ Finally - code!.mp4
5. The Joy of Code #5_ If-statements.mp4
6. The Joy of Code #6_ Random behaviour.mp4
7. The Joy of Code #7_ Lettuce is good for you!.mp4
8. The Joy of Code #8_ Keeping your code clean.mp4
9. The Joy of Code #9_ Snakes on a plane!.mp4
10. The Joy of Code #10_ It's done -- Play the game!.mp4
11. The Joy of Code #11_ Make it your own.mp4
12. The Joy of Code #12_ Fun with sound.mp4
13. The Joy of Code #13_ Structure of a class.mp4
14. The Joy of Code #14_ A first look at variables.mp4
15. The Joy of Code #15_ Object interaction (first encounter).mp4
16. The Joy of Code #16_ Adding a score counter.mp4
17. The Joy of Code #17_ Bouncy coloured balls.mp4
18. The Joy of Code #18_ Bouncy balls with mouse input.mp4
19. The Joy of Code #19_ Class methods vs instance methods.mp4
20. The Joy of Code #20_ Image manipulation - first contact.mp4
21. The Joy of Code #21_ Image manipulation - take two.mp4
22. The Joy of Code #22_ Refactoring for good style.mp4
23. The Joy of Code #23_ Animating the image.mp4
24. The Joy of Code #24_ Smoke and mirrors.mp4
25. The Joy of Code #25_ Dealing with errors.mp4
26. The Joy of Code #26_ How to display text_(720p).mp4
27. The Joy of Code #27_ Object interaction - a second look.mp4
28. The Joy of Code #28_ Return of the object interaction.mp4
29. The Joy of Code #29_ Viewer questions_ timers.mp4
30. The Joy of Code #30_ Introduction to loops.mp4
31. The Joy of Code #31_ More Loopiness.mp4
32. The Joy of Code #32_ Pretty pictures with while loops.mp4
33. The Joy of Code #33_ Playing Breakout! Collision detection.mp4
34. The Joy of Code #33_ Playing Breakout! Collision detection_(720p).mp4

VIDEO ON GAMER KID

1. 14-Year-Old Prodigy Programmer Dreams In Code.mp4

TEACHER COMMENTARY

1. The Joy of Code - Teacher Commentary 5_ Early coding examples.mp4
2. JoC - Teacher Commentary #10_ Motivation and reflection.mp4
3. JoC - Teacher Commentary #22_ Tracing control flow.mp4

Appendix A-7: Greenfoot Developer



- Michael Kölling Software Developer
- Michael Kölling is a professor and software developer with the School of Computing at the University of Kent. Originally from Bremen, Germany, he is also a key member of the team that developed the BlueJ and Greenfoot Java learning environments. [Wikipedia](#)

Appendix A-8.1.1: Teachers Rollout Course on Greenfoot

Creating Java Programs with Greenfoot

Overview

This workshop engages students who understand basic programming concepts to create Java programs. Participants use Greenfoot™—a free development environment from the University of Kent—to write standard Java syntax and create games, simulations, and applications that interest them. Along the way, students master object-oriented programming terminology and concepts such as procedure definition, algorithms, functions, compiling, debugging, and conditional statement execution.

Duration

- 16 hours

Target Audiences

Primary Audience

- College/university faculty who teach computer programming, information communications technology (ICT), or a related subject
- Secondary school teachers who teach computer programming

Secondary Audience

- Secondary schools teachers who seek to incorporate technology into their curriculum and awaken students' interest in computer science

Prerequisites

Required

- Basic understanding of at least one programming language
- The ability to follow software installation instructions and install Greenfoot on a computer

Suggested

- Getting Started with Java Using Alice

Suggested Next Courses

- Java Fundamentals
- Java Programming

Lesson-by-Lesson Objectives

Getting Started With Greenfoot

- Download and install Greenfoot
- Describe the components of the Greenfoot development environment
- Create an instance of a class
- Describe classes and subclasses
- Recognize Java syntax used to correctly create a subclass

Using Methods, Variables and Parameters

- Define parameters and how they are used in methods
- Understand inheritance
- Describe the properties of an object
- Examine the purpose of a variable
- Discuss programming concepts and define terminology

Appendix A-8.1.2: Teachers Rollout Course on Greenfoot

Working with Source Code and Documentation

- Demonstrate source code changes to invoke methods programmatically
- Demonstrate source code changes to write an IF decision statement
- Describe a procedure to display object documentation

Developing and Testing an Application

- Demonstrate program testing strategies
- Recognize phases for developing a software application

Using Randomization and Understanding Dot Notation and Constructors

- Create randomized behaviors
- Define comparison operators
- Create IF-ELSE control statements
- Create an instance of a class
- Recognize and describe dot notation

Defining Methods

- Describe effective placement of methods in a super or subclass
- Simplify programming by creating and calling defined methods

Using Sound and Keyboard Control

- Write programming statements to include sound in a program
- Write programming statements to include keyboard movements in a program

Creating a World, Animating Actors, and Ending a Game

- Construct a world object using a constructor method
- Create an object using a constructor
- Write programming statements to use the new keyword
- Define the purpose and syntax of a variable
- Recognize the syntax to define and test variables
- Write programming statements to switch between two images
- Write programming statements to end a game

Understanding Abstraction

- Define abstraction and provide an example of when it is used

Using Loops, Variables, and Strings

- Create a while loop in a constructor to build a world
- Describe an infinite loop and how to prevent one from occurring
- Use an array to store multiple variables used to create a world
- Create an expression using logic operators
- Describe the scope of a local variable in a method
- Use string variables to store and concatenate strings

Putting it All Together with Greenfoot

- Apply your Greenfoot knowledge to create a Java game

To search and register for events scheduled in your area, visit the [Academy events calendar](#).

Appendix A-8.2.1: Rollout to WCED Teachers Workshop 1



Greenfoot Registrations

Venue: Durbanville HS

Friday 11 April and Saturday 12 April 2014. (see times below)

Programme:

Friday:

15h00 – 15h30: Registrations and Attendance register

Refreshments (finger goodies) for teachers as they arrive and whilst they register. Venue: Library.

15h30 -18h00: Greenfoot

No break for tea afterwards (leg stretch as per guidance from facilitator, Wilhelm Rothman).

Saturday:

08h00 - 08h30: Attendance register

Coffee/Tea on arrival

08h30: Class commences

(Be punctual please)

10h30 -11h00: Tea break

13h00-14h00: Lunch Break

(May be shortened on discretion of facilitator)

14h00 – 16h30: Class resumes

(No break after lunch)

Appendix A-8.2.2: Rollout to WCED Teachers

The following teachers are registered for Greenfoot:

ID	Surname	Name	School	District	Email	Greenfoot
1			School of Hope	Metropole_Central		TRUE
2			Uitzig Sec	Metropole_North		TRUE
3			Bosmansdam HS	Metropole_North		TRUE
4			Bosmansdam HS	Metropole_North		TRUE
5			Knysna HS	Eden Central Karoo		TRUE
6			Mountain View Academy	Metropole_North		TRUE
7			Trafalgar HS	Metropole_Central		TRUE
8			Durbanville HS	Metropole_North		TRUE
9			New Orleans Sec	Cape Winelands		TRUE
10			New Orleans Sec	Cape Winelands		TRUE
11			Vredenburg HS	West Coast		TRUE
12			Heritage College	Metropole_Central		TRUE
13			WCED	Metropole_Central		TRUE
14			Worcester Gymnasium	Cape Winelands		TRUE
15			Durbanville HS	Metropole_North		TRUE
16			Brackenfell HS	Metropole_East		TRUE
17			Bernadinho Heights	Metropole_North		TRUE
18			Vuyiseka Sec	Metropole_South		TRUE
19			*Asibambisane HS	Metropole_North		TRUE
TOTAL:						19

NB:

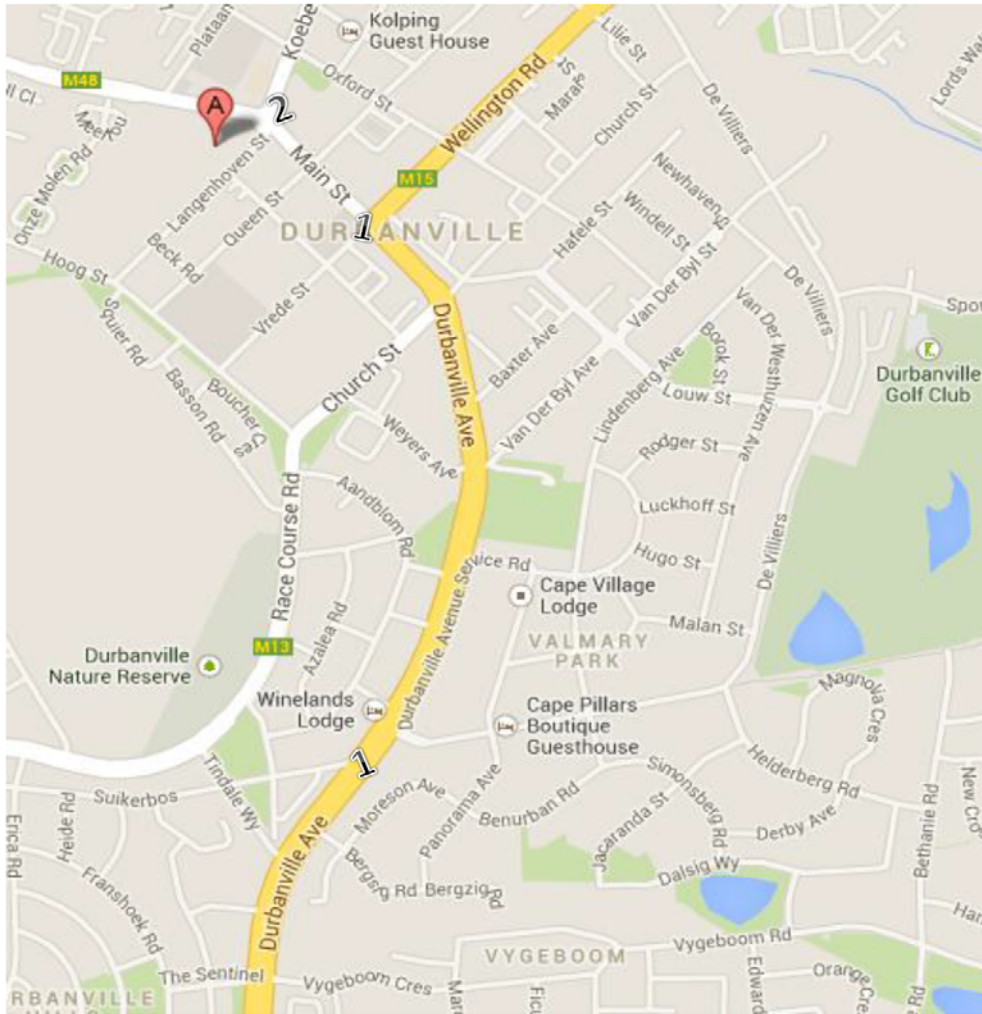
- Confirmation is still sought from a few teachers.
- No more than 25 teachers will be registered for this course.
- No accommodation has been arranged for individual teachers. Teachers who need accommodation must let me know so that I can make the necessary arrangements.
- I have attached the Z43 (transport form) and claim form hereto. Teachers need to print the documents, complete and submit to me on the Saturday after lunch.
- Please see below for directions to Durbanville HS.

IGHSAAN FRANCIS

DEPUTY CHIEF EDUCATION SPECIALIST: CAT and IT

Appendix A-8.2.3: Location sent to Teachers

Directions to Durbanville HS:



1. Establish a link to the N1 from either the Southern Suburbs, (via R300, M5 or N7) or Northern Suburbs (via Okavango, Brackenfell Blvd, Platteklouf Rd, etc.)
2. If approaching from south (Cape Town side) take the left off-ramp from N1 onto Durban Rd. Continue left towards Tyger Valley Centre. If approaching from the North (Paarl side) take left off-ramp from N1 onto Durban Rd. Proceed to your right into Durban Rd and continue towards Tyger Valley Centre.
3. Proceed with Durban Rd (which will become Durbanville Ave (#1 on the map) then Main St) till you get the round-about / circle (#2 on the map).
4. Take the first exit out of the round-about/circle.
5. Your destination (A) will be on your right.

Appendix A-8.3.1: Rollout to WCED Teachers Workshop 2



Greenfoot

Venue: Bosmansdam HS

Friday 30th and Saturday 31st May 2014.

Provisional Programme:

Friday:

14h00 – 14h30: Registrations and Attendance register

Refreshments (finger goodies) for teachers as they arrive and whilst they register.

14h30 -16h00: Greenfoot

16h00 – 16h30: Tea break

16h30 – 18h00: Greenfoot resumes

Saturday:

08h30 – 09h00: Attendance register

Coffee/Tea on arrival

09h00: Class commences

(Be punctual please)

10h30 -11h00: Tea break

11h00 – 13h00: Greenfoot resumes

13h00-14h00: Lunch Break

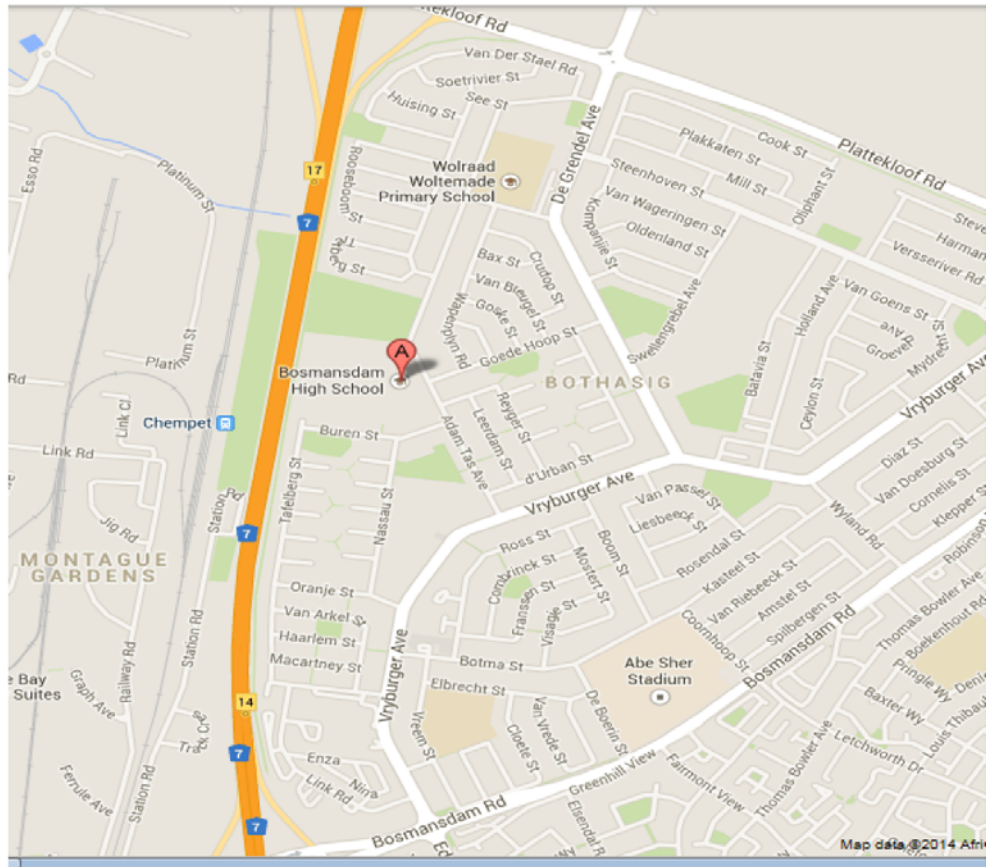
(May be shortened on discretion of facilitator)

14h00 – 16h30: Class resumes

(No break after lunch)

Appendix A-8.3.3: Location of WCED Teachers Workshop 2

Directions to Bosmansdam HS (Bothasig):



1. Establish a link onto the N7 from either the N2 or N1.
2. Continue onto N7 towards Malmesbury
3. Take exit 14 for Bosmansdam Road/Bosmansdamweg toward M8/Milnerton
4. Keep right at the fork, follow signs for M8/Edgemoed/Bothasig
5. Turn right onto Bosmansdam Rd
6. Turn left onto Vryburger Ave
7. Turn left onto Adam Tas Ave

Destination () will be on the left

Appendix A-8.3.4: Timesheet o.b.o. Oracle for WCED Teachers Training Workshop 2



Oracle-12273

Week Ending:	31 May 2014
Assignee Name:	Wilhelm Rothman
Company Name:	Oracle
Department:	IT
Start Day:	30/05/2014
Mobile No:	082 774 1645

Weekday	Date	Start Work	Time Out (Lunch)	Time In (Lunch)	End Work	Total Hours (including Lunch)	Total Normal Hours (Excluding Lunch)	Daily Norm Hours
Monday	26/05/2014							
Tuesday	27/05/2014							
Wednesday	28/05/2014							
Thursday	29/05/2014							
Friday	30/05/2014	12h30	15h10	15h25	18h00	5.5	5h00	5.5
Saturday	31/05/2014	08h30	17h30	13h15	13h30	9	8h45	9

WEEKLY TOTAL

Total Hours:	14.5	
Normal hours:		
Overtime hours:		

I, hereby certify that the above hours and travelling details (if applicable) are a correct record of time for the independent contractor named above.
As this document forms the basis of Invoice and in terms of the requirements of the S.A.R.S the onus is on both the client and the independent contractor to each keep copies of the timesheet.



INDEPENDENT CONTRACTOR'S NAME

31/5/2014

DATE

CLIENT AUTHORISING SIGNATURE

FULL NAME OF CLIENT SIGNATURE

Appendix A-8.3.5: WCED Teachers Training Signatures Workshop 2

ID	Surname	Name	School	Email	30-05-2014	31-05-2014
1			Fezeka Sec			
2			De la Bat School			
3			Zwaanswyk HS			
4			Eben Donges HS			
5			Cloetesville Sec			
6			Belville South Sec			
7			The Oracle Academy			
8			Crestway Sec			
9			St Andrews Sec			
10			CAMST			
11						
12			Sarepta Sec			
13			St Cyprians			
14			The Oracle Academy			
15			Eben Donges HS			
16			Bosmansdam HS			
17			CPUT			
18			Bernadinho Heights HS			
19			DEDAT			
20			Masibambisane HS			
			Simelane HS			

Appendix A-8: Oracle Instructors Certificate

Please see Physical Files at Pool of Resources.

Appendix A-9: Emails sent to Authors of the APOS Theory

Appendix A-9.1: Email 1 to Prof Dubinsky (12 February 2015)

>
> -----Original Message-----
> From: Edward Dubinsky [mailto:edubinsky@math.umd.edu]
> Sent: Thursday, February 12, 2015 5:37 PM
> To: Wilhelm Rothman
> Subject: Re: Framework
>
>
>
> Wilhelm,
>
>
>
> See below for responses.
>
>
>
>
>
>> At the moment I am investigating the TRIAD on intra-, inter- and
>
>> trans- stages as explained in Chapter 7. I must admit that when
>
>> assimilations of new Constructions into existing Schemas are
>
>> considered plus the interaction of Schemas, things become a bit
>
>> tricky, esp. when two triads are joined,
>
>> esp. when the intra-xx is looked at from an intra-yy, inter-yy and
>
>> trans-yy stage. I think it is called a double triad when two schemas are
> combined.
>

Appendix A-9.2: Email 2 from Prof Dubinsky (15 March 2015)

I think you might do better by contacting some other people knowledgeable about APOS Theory and discuss things with them, as well as me, by email and Skype. Here are some people you can contact --- and you tell them that I suggested you contact them.

Jim Cottrill <[mailto:jcottrill@math.umd.edu]>
Kirk Weller <[mailto:kirkweller@math.umd.edu]>
Maria Trigueros <[mailto:trigueros@math.umd.edu]>
Ilana Arnon <[mailto:ilana@math.umd.edu]>
Draga Vidakovic <[mailto:dragavid@math.umd.edu]>

Appendix A-9.3: Email 2 from Prof Dubinsky (21 March 2015)

Wilhelm Rothman

From: Edward Dubinsky <...>
Sent: Saturday, March 21, 2015 3:32 PM
To: Wilhelm Rothman
Subject: Re: Research in Cape Town

Wilhelm,

You should write Cottrill again. I know he had a very busy period which may have eased up now. He mentioned to me recently that he has a version of ISETL written in Java. You might be interested to have a look at it. He says it works well for him.

The book was a true, collective effort and no single author was the "brains" behind it. We tried to express this fact by listing the authors in alphabetical order.

Best wishes for the onset of Spring.

Ed

Appendix A-9.4: Email to Dr Illana Arnon

From: Wilhelm Rothman [mailto:rothmanw@Telkomsa.net]
Sent: Wednesday, March 04, 2015 9:09 AM
To:
Subject: APOS Research in Cape Town
Importance: High

Dear Prof Arnon

I got your address from prof Ed Dubinsky after communicating with him for some time now on my research. He actually gave me a number of addresses linked to the APOS Theory book you released in 2014 regarding all authors. Your name seems to be mostly prominent within the book and hence my email.

My research reached a point now where I need to "compare" my conception to the concept regarding my interpretation of a programming language and the work you have done within APOS Theory from a mathematical perspective. Thank goodness for the book because otherwise all the articles on APOS floating around can cause more confusion than good.

In a nutshell:

My aim is to link the art of programming into the APOS Theory which is mainly mathematically driven, but in South Africa we have a major problem in that our learners do not take normal mathematics in their numbers anymore or do badly in their final matric year when good marks are necessary for disciplines such as engineering, medical profession etc. that the country is suffering in the form of a brain drain. I must comment that here in Cape Town many chefs or recipe/ingredient mixers have seen the light in mathematics, where learners are given recipes and do well based on (my interpretation) an action based approach and process and object never realize, if I can use APOS terminology. These cooks are also only available to those who can afford such an experience and they are mainly old ex teachers that taught at a school and has insight into previous papers.

My situation

I am involved in teaching and the IT world from an educational and business perspective at CPUT (Cape Peninsula University of Technology), but my passion has been with the value that programming can bring to the table. "Recently" programming languages evolved to such an extent that the red tape of syntax is either minimized or balanced through visual programming interfaces accessible to all.

My specialization fields are Internet Programming in proprietary and open source, such as ASP.NET and PHP, LINUX etc. I am also lecturing Java as an object oriented prog language and acquired instructors status from Oracle in Alice,

Appendix A-9.5.1: Email Response from Dr Arnon (8 March 2015)

Wilhelm Rothman

From: Arnon <[redacted]>
Sent: Sunday, March 08, 2015 12:21 PM
To: rothmanw@Telkomsa.net
Subject: RE: APOS Research in Cape Town

Dear Wilhelm Rothman,

Thank you for inviting me to work with you. Let me see if I understood your intention. Did you mean to apply APOS theory to the teaching of programming languages? It happens often that people ask us whether APOS is applicable to other areas of knowledge. Our answer to that is that this question requires research. My opinion is that it is. Alas, my knowledge does not cover programming, hence cannot contribute to your work. Any other questions regarding APOS will be most welcome.

Sincerely yours,
Ilana Arnon.

Appendix A-9.5.2: Email Response from Dr Arnon in attached Word document 04-12-2015 @ 8:44 pm

Wilhelm, hello

I see two issues in your letters: one concerns the issue of concrete versus formal operations according to Piaget, the other about GD in the fractions research.

The GD issue is easier so I will start with it.

First notice that the research on fractions took place as early as 1991, long before the publication of the paper by Asiala and al.¹, where the format of an APOS research including the format and role of a GD was formally addressed. Never the less, we did have an (oral) GD: We spend several years in classrooms watching the children struggling with fraction activities, combined these observations with Piaget's and APOS theories, and our knowledge of mathematics. All these were combined into an unofficial GD of the time; As a result of this experiment further versions of the GD and corresponding materials were produced and tested

Now for the second question.

Although Piaget (translated from French) spoke of concrete versus formal operations, I prefer to discuss the issue using the terms concrete versus Abstract, which have many different "definitions" in the literature. The "definitions" I use are explicitly explained in the book, as well as in a paper on fractions as equivalent classes². According to these meanings, children (most of them, some know better) of what Piaget names "the age of concrete operations" need to start with activities on concrete objects and are able to develop out of them abstract objects. Students of "the age of formal operations" can start with abstract activities on abstract objects, and produce out of them new abstract objects. I guess that for your students, abstract activities on abstract objects would mean simple programs such as offered by the software ISETLE recommended in many of the APOS related research reports as well as in the book. I would not know what would be abstract objects in your subject matter.

Is this helpful?
Ilana

¹ Asiala, M., Brown, A., DeVries, D., Dubinsky, E., Mathews, D., & Thomas, K. (1996). A framework for research and curriculum development in undergraduate mathematical education. *Research in Collegiate Mathematical Education II, CBMS* 6, 1-32.

² Arnon, I., Neshet, P., & Nirenburg, R. (2001). Where do fractions encounter their equivalents? Can this encounter take place in elementary school? *International Journal of Computers for Mathematical Learning*, V. 6(2), pp. 167-214. Kluwer Academic Publishers, The Netherlands

Appendix A-9.6: Western Cape Education Plan to Salvage Mathematics

A plan to encourage more learners to take Mathematics and Physical Sciences and improve learners' performance in these subjects was already paying off with learners at a Mitchells Plain High School begging their teacher to stay in during break to do Mathematics.

Phaldie Tregonning, deputy principal and Mathematics teacher at Rocklands High School, said the learners were especially enthusiastic about using the notebook computers provided by the Western Cape Education Department (WCED).

The investment in ICT was part of the WCED's Mathematics and Science Strategy to increase Grade 12 Mathematics passes from 15 000 to 22 000 in 2019 and increase Physical

Sciences passes from 11 500 to 16 000 in the same period.

The department announced that it was investing approximately R113 million in the next five years in the strategy.

The department budgeted R12.4 million for the current financial year.

Each of the 48 Dinaledi Mathematics and Science schools would receive notebooks onto which digital resources should be loaded to support the Mathematics curriculum, and a laptop trolley. Once the roll-out was complete at the Dinaledi Main Schools, the new technology would also be made available to the 24 Dinaledi Incentive Schools in the province.

The schools also received textbook top-ups and maths and science kits and would benefit from the refurbishing of ICT and science laboratories.

"Proficiency in Mathematics forms the basis of many of our further education and training fields in science, technology, engineering and business - all key sectors for a growing economy such as ours.

Therefore, as a government, we are working hard to encourage more learners to take Mathematics and science so that we can grow our future economic workforce and broaden the life chances of our learners," said Western Cape Education Minister Donald Grant.

He visited Rocklands Secondary School where Tregonning demonstrated the use of ICT in enhancing Mathematics tuition in the classroom.

Tregonning used the graphing programmes extensively as part of his teaching to illustrate functions and graphs. He said in the past learners simply did not answer questions on graphs in the exams.

Brian Schreuder, Deputy Director-General: Curriculum and Assessment Management, said the Mathematics and Science strategy was relevant to all high schools and was linked to the department's Literacy and Numeracy Strategy which focused on foundational learning.

“The approach to schools is a staggered one. Once we create centres of excellence in Mathematics and Science in these (Dinaledi Main and Dinaledi Incentive) schools, other schools will be targeted over a five year period.”

He said from January 2013, the department wished to see more learners taking Mathematics in Grade 10. There were many learners doing Mathematical Literacy who could cope well with Mathematics and schools were asked to identify those learners and advise them accordingly.

The department expected an improvement in the quality of Grade 12 passes in terms of the average mark in Mathematics and Physical Sciences and the number of learners who get distinctions and B and C symbols. Schools that meet their targets would receive incentives such as funding to buy additional resources for the school.

Appendix B-1: Intervention 1 (Abstraction [Abstract Thinking] Assessment)

Abstraction: Please make an X in the block of your choice.

After the person stepped up and down for 239 times, will he/she be On the Floor, on Top of the step, or is your answer a GUESS



ON THE FLOOR	ON TOP OF STEP	MUST GUESS

Figure B-1: Abstraction Exercise 1

Appendix B-2: Questionnaire on Mathematics in General (Learner X)

MATHEMATICS

1. How do you see mathematics in your life at this stage?

a. Is it important to you?

yes, I need good marks so next year I can take it and
pure math

b. If you answered yes at a) how would you describe the importance in your life – your understanding of what mathematics mean in your life?

there are jobs that need math that I want
to maybe do

c. Are you struggling with mathematics?

a little some parts of it

d. What are you struggling with in mathematics at this point in time?

simplify exponents

e. When you add $14 + 4$, describe exactly what you do in your head to get to the answer?

I just add it up, I just do it

f. Draw a triangle as perfect as can be.



g. What is the sum of that triangle's angles in degrees?

$60 + 60 + 60$

h. How do you know that your answer at point g is correct?

I don't know if it's correct, just a guess

I don't have any assistants

Appendix B-3: Questionnaire on Mathematics in General (Learner Y)

MATHEMATICS

3:30 pm

1. How do you see mathematics in your life at this stage?

a. Is it important to you?

Yes.

b. If you answered yes at a) how would you describe the importance in your life – your understanding of what mathematics mean in your life?

Well it is very important, some of the maths I do not understand.

c. Are you struggling with mathematics?

Yes.

d. What are you struggling with in mathematics at this point in time?

geometry and factorisation.

e. When you add $14 + 4$, describe exactly what you do in your head to get to the answer?

I take 14 and I break it up
say $10 + 4 + 4 = 18$

f. Draw a triangle as perfect as can be.



g. What is the sum of that triangle's angles in degrees?

unknown at this stage.

h. How do you know that your answer at point g is correct?

NO extra classes.

Appendix B-4: Questionnaire on Mathematics in General (Learner Z)

MATHEMATICS

3:28pm

1. How do you see mathematics in your life at this stage?

I see mathematics as an obstacle I'm struggling to get over.

- a. Is it important to you?

Yes

- b. If you answered yes at a) how would you describe the importance in your life – your understanding of what mathematics mean in your life?

Very important as not only would I like to get into a good university but I think it would make other tasks in my life easier.

- c. Are you struggling with mathematics?

Yes.

- d. What are you struggling with in mathematics at this point in time?

Algebra.

- e. When you add $14 + 4$, describe exactly what you do in your head to get to the answer?

I add $4 + 4$ together first then use the 1

- f. Draw a triangle as perfect as can be.



- g. What is the sum of that triangle's angles in degrees?

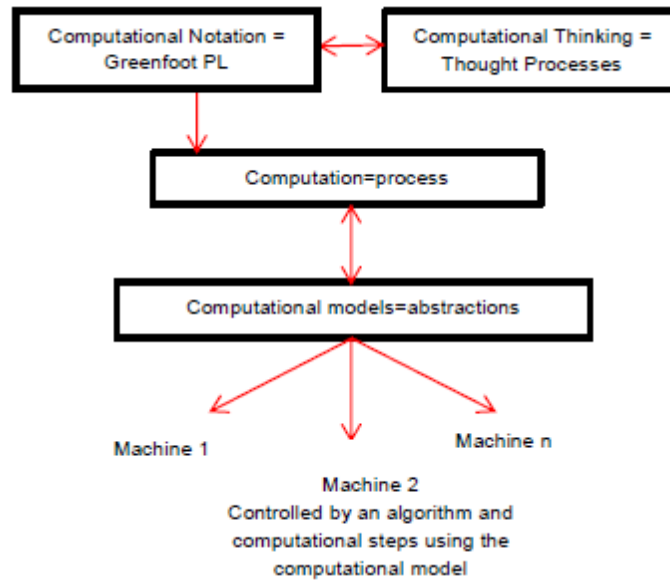
180°

- h. How do you know that your answer at point g is correct?

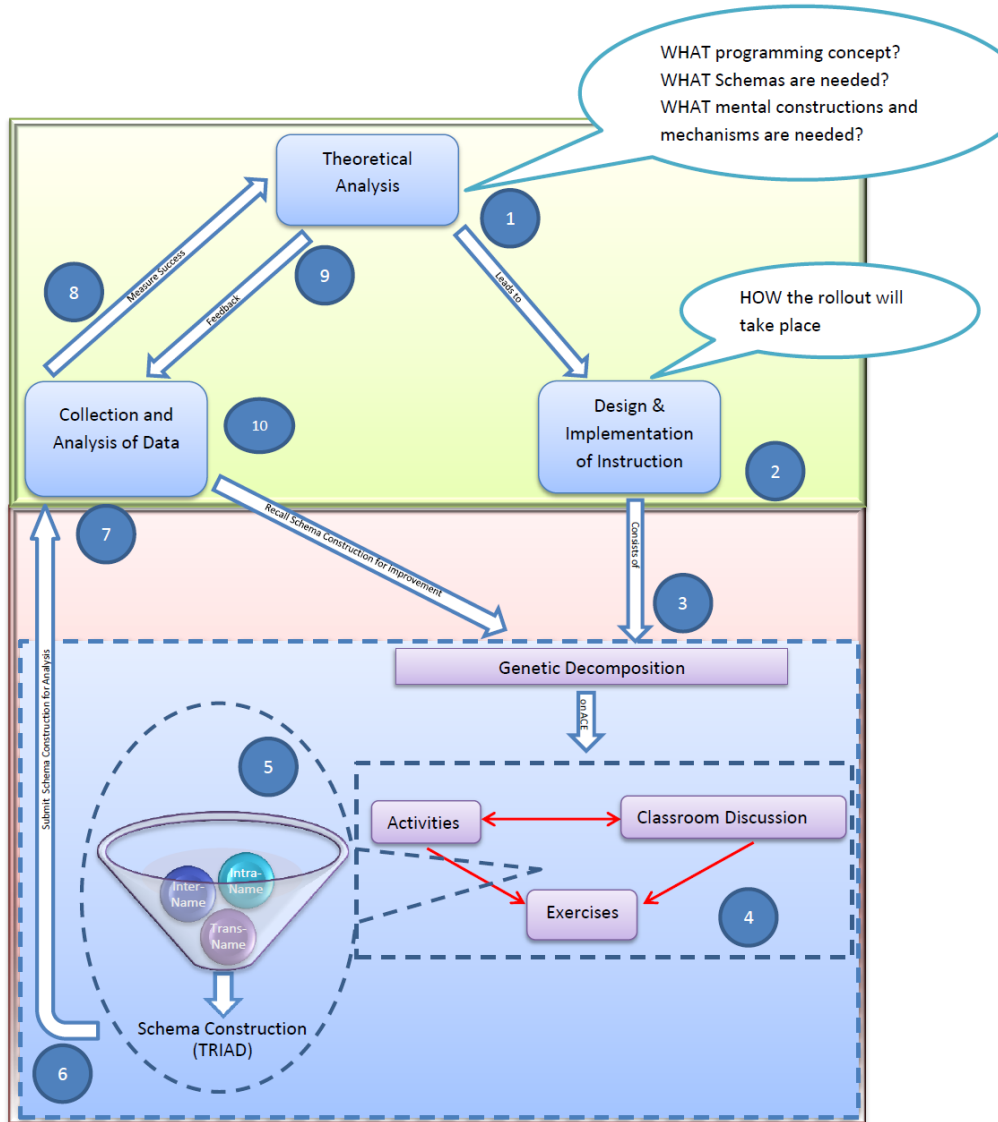
I did it in class.

I go to extra classes when needed (maths teacher at school)

Appendix C: Intervention 2A-1 computational thinking in motion (Compiled interpretation taken from Denning (2017) and AHO(2011))



Appendix D-1: Intervention 2A-2 (Genetic Decomposition Process adapted from Arnon et al., 2004)



Appendix D-2: Intervention 2A-3 (Genetic Decomposition of “Load a Greenfoot Scenario”)

Preliminary Genetic Decomposition

1P: **The Action:** Viewing videos illustrating the tools necessary for Greenfoot programming, i.e. scenario - World, Actor. Physical clicking the option step-by-step as illustrated in the video.

2P: **The Process** of interiorizing the Action to activate Greenfoot and using Act or Run to execute a scenario without looking at the video again.

3.1P: The Action of opening the Greenfoot editor.

3.2P: The Action watching the video on a basic Greenfoot scenario within the context of webpage.

3.3P: The Process is created by interiorizing the steps of such a scenario.

Refinement

1.1R: The learner logs into the Moodle site and familiarise him/herself with such an LMS. It is just a matter of clicking on the hyperlinks to activate the videos.

1.2R: The learner forms part of an activity, also known as a flipped classroom technique, where he/she watches a video on the implementation of these tools and follow instructions step-by-step.

1.3R: The learner takes action by opening Greenfoot. Revisit the video if he/she gets stuck.

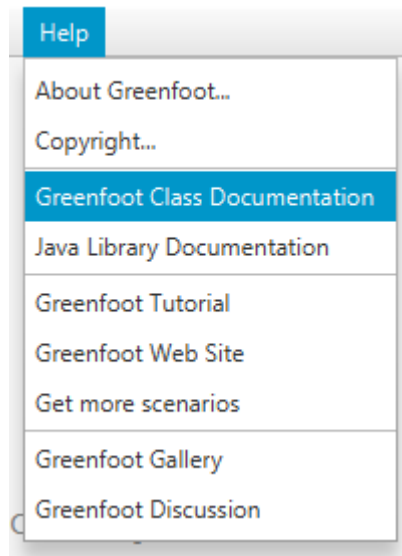
2.1R: Open Greenfoot. Check for necessary vital signs. Load any scenario in downloads and click on Act and Run.

3.1R: The learner opens Greenfoot and checks for the vital signs such as World, Actor, and menu items.

3.2R: The learner mimics the video and runs a scenario.

3.3R: Now create a scenario. Save the scenario at some location with a different name. No video was used

Appendix D-3: Intervention 2B (Help Documentation in Greenfoot)



All Classes

- Actor
- Color
- Font
- Greenfoot
- GreenfootImage
- GreenfootSound
- MouseInfo
- UserInfo
- World

PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

Package greenfoot

Class Summary

Class	Description
Actor	An Actor is an object that exists in the Greenfoot world.
Color	A representation of a Color.
Font	A representation of a Font.
Greenfoot	This utility class provides methods to control the simulation and interact with the system.
GreenfootImage	An image to be shown on screen.
GreenfootSound	Represents audio that can be played in Greenfoot.
MouseInfo	This class contains information about the current status of the mouse.
UserInfo	The UserInfo class can be used to store data permanently on a server, and to share this data the Greenfoot web site.
World	World is the world that Actors live in.

Appendix D-4: Intervention 14E (Genetic Decomposition of IF statement)

Preliminary Genetic Decomposition (IF Statement)

1P: **The Action:** Viewing videos illustrating if statement looking at Moodle using Activity 4 (Action on Object (Rocket). Confirmation can also be found in "Joy of Code Video #5: If-statements" within same Activity 4.

2P: **The Process** of interiorizing the Action of using the IF statement can be found in running the video "Actions on Object (Rocket) and Joy of Code Video 5..

2.1P: The Action of watching video on "Creating a Scenario" which forms part of GD

2.2P: The Action taken to watch the video 5 on Joy of Code, followed by video Actions on Object (Rocket).

3P: The Process is created by interiorizing the steps of such an IF statement in the Greenfoot scenario.

3.1P: Create the code in scenario by memorising the code lines.

3.2P: Use abstraction by embedding code into `turnAtEdge()`.

Refinement

1.1R: The learner logs into my Moodle site on wrru.co.za/moodle and familiarise him/herself with such an LMS. It is just a matter of clicking on the hyperlinks to activate the videos.

1.2R: The learner forms part of an activity, also known as a flipped classroom technique, where he/she watches a video on the implementation of the IF concept.

1.3R: If the learner is still at Action phase the video "Creating a Scenario" can be run to help the learner in creating a scenario discovering the Rocket's positional coordinates when positioned left and right in the world.

2.1R: Create a new scenario in Java and create a World and a Rocket Actor that will live in that world. Compile the scenario and run to watch rocket move in one direction.

2.2R: The learner mimics the code just for left side only, namely `if (getX() < 20) turn(90);`

2.3R: The learner rollout the method to embed the if statements just for the right hand side namely `if (getX() > 580) turn(90);`

3.1R: Now create a Greenfoot scenario without looking at those steps from some external medium, such as video or textbook and combine both left and right hand side conditions. Save the script at the correct location.

3.2R: The learner create `turnAtEdge()` method and embed the code within. Create a call to that method in the `act()` method.

Appendix E-1: Intervention 3A (Introduction of the Moodle LMS)

CURRO

PREREQUISITES

WEEK 1

PREREQUISITES

1. Headphones to listen to Videos
2. Microphone/Skype tool to create own sounds
3. PC to create applications.
4. Explain to each other how code/theory works
5. Always do your own work and ask for help when needed
6. No real homework apart from watching videos
7. No race or competition.
8. <http://moodle.efundo.co.za>
9. Write down list of names

Appendix E-2: Intervention 3B (Juggling enactment to enforce Moodle usage among learners)

JUGGLING

CLASS GROUP:

NAME:

Before your evaluation tick the following textboxes

I watched the video on basic juggling

I practiced the methods illustrated in the video at home

I can Juggle with ONE ball in a linear movement (tossing ball up and down)

I can Juggle with ONE ball in circular movement (tossing ball up and down)

I can Juggle with TWO balls, using two hands (tossing ball and handing over to other hand)

I can Juggle with TWO balls, using one hand ONLY (tossing TWO balls with one hand)

I can Juggle with TWO balls, using one hand ONLY (tossing TWO balls in all directions)

I can Juggle with THREE balls (tossing THREE balls with two hands)

If you could not tick all the boxes, what do you think must be done for you to be able to tick every box?
Please state clearly how you will accomplish this task.

What did you see as a problem to accomplish this task? E.g. Self-motivation, no Internet access etc.

If you could tick all the boxes, why do you think you had success?

Appendix E-3: Intervention 3C (Moodle and Generalised Terminology)

CURRO

QUESTIONNAIRE

WEEK 1

QUESTIONNAIRE

(Answer on rear if you need more space)

1. Explain in your own words how would you solve any mathematics problem? What goes on in your mind when confronted by a maths problem. Even if it scares you then state that!
2. Explain the term "computer programming" - your understanding with an example.
3. Have you done comp. programming before? Y/N If Y(es) give me a brief background on what exactly did you do in programming. If N(o) state why not and if you think you cannot do it and why you think it's not for you.
4. Have you heard of the programming language Greenfoot before? Y/N If Y(es) tell me what it is or how you know Greenfoot.
5. What do you understand by the term "ABSTRACTION". Use any example to illustrate your understanding.
6. Have you been using a technique "flipped classroom" before. Explain what you understand by "flipped classroom".

Appendix F-1: Intervention 4A (Creating a Linux Server with external access)

GreenFoot Introduction for WCED Teachers



University of Kent LA TROBE

 [GREENFOOT TEACHERS ATTENDING COURSE](#)

This Quiz will help me to pitch my lecturing during the Greenfoot course. All questions are shown and you must click on NEXT and then SUBMIT to save all your answers. Much appreciated.

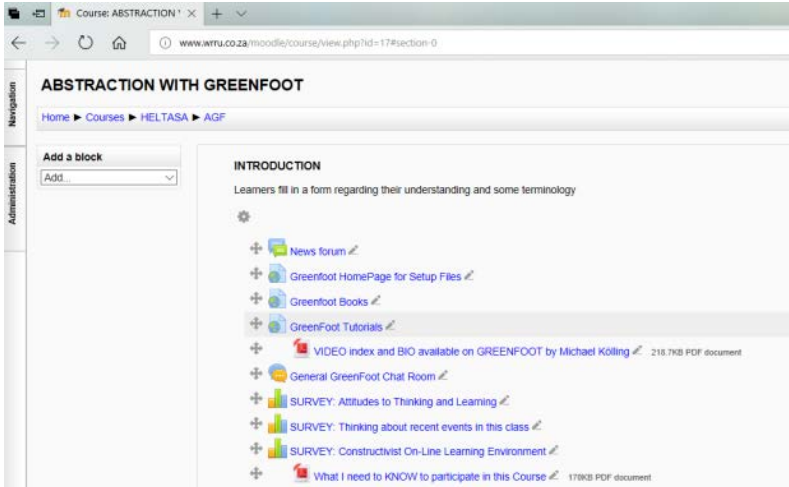
 [Overview of Course](#) 176KB PDF document

 [Joy of Code - Index - Videos by Michael Kölling \(Picture of Michael\)](#) 218.7KB

 [News forum](#)

 [Social forum](#)

Appendix F-2: Intervention 4B (Creating a Cloud-based Moodle LMS)



The screenshot shows a Moodle course page titled "ABSTRACTION WITH GREENFOOT". The page includes a navigation breadcrumb: Home > Courses > HELTASA > AGF. On the left, there is an "Add a block" section with a dropdown menu. The main content area is titled "INTRODUCTION" and contains a list of resources and activities:

- News forum
- Greenfoot HomePage for Setup Files
- Greenfoot Books
- GreenFoot Tutorials
- VIDEO index and BIO available on GREENFOOT by Michael Kölling (218.7KB PDF document)
- General GreenFoot Chat Room
- SURVEY: Attitudes to Thinking and Learning
- SURVEY: Thinking about recent events in this class
- SURVEY: Constructivist On-Line Learning Environment
- What I need to KNOW to participate in this Course (179KB PDF document)

Figure 4.15: Moodle on Abstraction and Greenfoot

Appendix G-1: Intervention 5A (Introduction to Greenfoot)

LESSON 1

1 HOUR

Getting Started with Greenfoot

1. State components to download and the website greenfoot.org
2. Ensure everybody has Book Scenario (down)loaded.
3. Sketch the aim of Greenfoot.
4. Introduce <http://rothman.for-better.biz:80980/moodle>
 - a. Give brief walk through
5. Select any one of scenarios:
 - a. Wombat
 - b. Little crab
 - c. Asteroids
6. Run through slideshow.
7. Do exercise on Lesson 1: Getting Started with Greenfoot and know all Greenfoot components/terminologies.

Vocabulary as published by Oracle

Directions: Write the definition for each vocabulary word below.

1. More specific types of a class.
2. The template that defines the substance of an object, such as its appearance, features, and movement.
3. The more generic, overarching class of a group of classes.
4. An object of the class.
5. Translates the source code into a machine code that the computer can understand. This ensures that you added the source code or class correctly before you proceed.
6. Defines what all instances of each class are capable of doing. The behavior of each instance is determined by the source code of its class.

Appendix G-2: Intervention 5B (Revisit previous Activities)

HOMEWORK TICK EVERY ASPECT WHEN DONE

28-07-2014

1. Log into website: <http://moodle.efundo.co.za>
2. Login details: *c+surname+init* example for Adriaan Alexander calexandera. Use your own details!

Password: *C1111#c+surname+init* example C1111#calexandera
3. Please view Videos #1, #2, #3 in Joy of Programming by Michael Kolling. We will now program!
4. What is "abstraction" in general? How? Ask an older person and explain concept to him/het. Think of art, underground train maps, music demo and simplification in mathematics, e.g. $\frac{8x^3}{2x}$ and $\frac{8}{2x}$
5. Read definition of abstraction by Hazzan

The quality of your relationship between the object of thought and the thinking person. The more connections that person forms to the object, the more concrete he/she feels to it. How many connection can you make with the object?

6. Read about the APOS theory on website. What does APOS stand for?

Action = a transformation is and action when it is a reaction to external stimuli. Learner must construct actions during instruction based on activities given. (Look at video on juggling - external stimuli and analyse the process identifying actions for the process.)

Process(es) = When the learners reflects on an action scheme and interiorise it, it becomes process conception. It becomes part of the individual and he/she establishes control over it. (Tossing of the ball in a pattern)

Object = When an individual reflects on operations applied to a particular process, becomes aware of the process as a totality, realizes those transformations (actions or processes) and can act on them and can construct these transformations - the learner then think of the process as an object. (Basic Juggling as a unit/object)

Schema = Once constructed, objects and processes can be interconnected in various ways. A collection of processes and objects can be organised in a structured manner to form a schema. Schemas can be treated as objects and be included in the organization of "higher level" schemas. (Complex juggling)

Example: Mathematics e.g. $\frac{8x^3}{2x}$ and $\frac{8}{2x}$ Simplify the fractions and use the APOS theory. Write down APOS and show your understanding of the simplification within APOS.

7. Try to "juggle". View the video on website or download in class from shared drive and use APOS understand juggling. How can you use APOS to juggle. Where will you find Objects and schemas in "juggle". Write down every step. You will probably only reach actions and the process, but with practice you can get to the "juggling" as an object! Check out my steps on the website.

Thanks! Hopefully you enjoyed it - keep watching the videos. WcR

Appendix H: Intervention 6 (Applying Process and Object within mathematics)

DISCUSSION ON SIMPLIFICATION

28-07-2014

1. Let's examine $\frac{8x^3}{2x}$

Write down your answer after simplification.

SCHEMA name: Simplification.

Consists of prior elements:

ACTIONS (activities on how to identify factors and GCF & rules on working with exponents) and

PROCESSES which is when these actions are interiorized and

OBJECTS is then how you reflect on operations applied to a particular process and then becomes aware of the process as a totality (factors and GCF) and

SCHEMA is when one interconnects objects and processes in various ways to achieve *simplification*.

What does one need to know? (prior knowledge):

- a. Do you still need to think about the factors of the numerator and denominator and then about the **Greatest Common Factor** that exist in both numerator and denominator. If so then know your 8 Times table and go back and interiorise that. What do we mean by interiorize or internalise?
- b. The added variable (unknown) to the simplification is the concept of exponents. **Your prior knowledge:** when are exponents added/subtracted or multiplied/divided? The concept must be interiorised and if not go back to the drawing board.

2. Let's examine $\frac{8}{2x}$

Rewrite the fraction as a fraction broken up into factors, i.e. $\frac{2 \cdot 4}{2 \cdot 1 \cdot x}$

Find the GCF and remove.

Rewrite the fraction as a linear term, i.e. $4 \times ? \times ? \times x$?

Thanks! Hopefully you enjoyed it - keep watching the videos. WcR

Appendix I: Intervention 7 (Greenfoot as Process and Object)

LESSON 1: PROGRAMMING (Answer question on paper)

28-07-2014

1. Open Greenfoot and create a NEW scenario. Call it MyFirstProg. According to the videos that you watched write down the components you identify within the interface.
2. Create a "World subclass" and call it "MyWorld" and use Sand.jpg as background. Also create an "Actor subclass" and call it Wombat and use the wombat.png as image.
3. What must one do to make it ready to "work"? Do it and add an instance into "MyWorld". How would you add multiple instances into "MyWorld"?
4. The two buttons "Act" and "Run". What do they do? Discuss with your neighbour and write down your answer in your book given.
5. Write code to make the wombat move 5 paces at a time when the "Act" button is clicked. Click the "Run" button and explain to your neighbour the difference between the "Run" and the "Act" button by tracing the code.
6. Write down the acronym APOS in vertical format. Read the definition and let's discuss if you find any of the elements of A P O S in the activity above thus far?
7. Discuss how you can turn all of the above into a *process*. Decide on some action between you and your neighbour, describe it and then make it a process. **How will you do that?** Do another if you feel it is not yet a process according to your understanding of APOS. When it becomes a process call me, the teacher and explain why it's a process now. **Describe your process in book given. Use first person when describing your experience.**
8. Let's go back to $\frac{8x^3}{2x}$ and $\frac{8}{2x}$ The **object here is called simplification of fractions** and the schema is called simplification. Why would I call it an object but then just simplification a schema? **Write down all the tools and actions needed to perform the object (simplification) identified. Use the textbook given.**

Thanks! Hopefully you enjoyed it - keep watching the videos. WcR

Appendix J Intervention 8: Rollout of code in Greenfoot in Figure 4.20

LESSON 5: Revisit Concepts 25-08-2014

1. **Access site:** <http://moodle.efundo.co.za> **Your login:** c+surname+initial
Password: C1111#c+surname+initial, e.g. Login: *calexanderk* for Kristen Alexander and password will be *C1111#calexanderk*

2. **Write down the word (acronym) APOS vertically in the book given.**

State the meaning of each letter. Use an example to explain the meaning of each letter from normal activity, or in Greenfoot or in mathematics.

3. **Open the Greenfoot application and answer the following questions on paper**

First test the action in Greenfoot and then write down your answer!!!!

3.1 Where can you see how many classes make up the Greenfoot package?

3.2 How many of these classes do you count? Write them down.

3.3 What does a class consists of?

3.4 **A method isKeyDown().**

3.4.1 In which class will you find this method isKeyDown()?

3.4.2 What is the return type (which value does it return)?

3.4.3 What is the purpose of this method?

4. **Create a scenario**

Use the "Board.jpg" image and create a World by default. The default world dimensions are `super(600, 400, 1);`

4.1 Where can one see the instruction to create the world?

4.2 What does this mean? Count the squares. Write down your understanding.

4.3 What should the dimension be to represent a chess board? Think how many squares by counting the pieces on a chess board. Each square has four blocks in them already.

5. **You have chosen a dimension in previous question. Now test the x and y coordinates to coincide with positions of the chess pieces.**

5.1 Where can one set the chess board dimensions so we have an 8 x 8 and coordinates must be 0 to 7 on the x-axis and 0-7 on the y-axis?

5.2 Add an actor e.g. a Turtle in the world and position it on some black or white square. Now how can one determine what the x and y coordinates are?

6. Homework: Look at the video # 7. Enjoy!!

Appendix K: Intervention 9 (Making decisions towards Encapsulation)

LESSON 2: PROGRAMMING - Video 4 & 5 (Answer all questions on this page) 11-08-2014

1. Study Lesson 1 (handout) and answer all the questions on flip side of that page.
2. Look at Joy of Code#4 video. Put a turtle object in your world. How can one view all the methods of the turtle? Illustrate your action to your friend as well.
3. Position the turtle anywhere in the world and see if you can find the coordinates of the turtle using the available methods. What do you understand by coordinates? Write down your understanding, make a drawing of the world and show where coordinates (0,0), (600,400) and (100,300) are.
4. Look at my code how the turtle swing around when it reaches the end of the world. Use your drawing of the world and show to your friend where the turtle may fall off at the end of the world. Explain mathematically what the turtle must do NOT to fall off but turn around before it falls off. Look at my code at Question 5!
5. Open the editor of the turtle and add code inside the **act()** method's brackets. Every line must conclude with a semicolon ";". Add the code and run.

```
public void act()
{
    move(10);
    if(getX() < 20 || getX() > getWorld().getWidth() - 20)
        turn(10);
    if(getY() < 20 || getY() > getWorld().getHeight() - 20)
        turn(10);
}
```

6. Again you and your friend think about why this happens and explain on paper why the code makes the turtle turn around.
7. Let's see where **APOS** fits in.

A = ACTIONS	A command or code to do something e.g. turn(10) or move(10)
P = PROCESS	A sequence or number of steps/code to accomplish something
O = "OBJECT"	A name given to describe such a PROCESS
S = SCHEMA	A number of interrelated PROCESSES to form new SCHEMAS

Video #5 illustrates how the code can be encapsulated (covered or nested within) in a method. A name is then given to this set of actions (sequence of code (commands) to be executed or a process) which forms an "object" in your mind called **atWorldEdge()**. This method returns either a true or a false. A method always has a return type. If nothing is returned the return type is called **void**.

Thank you. - Hope you enjoyed this. Remember to watch videos 5 and 6! WcR

Appendix L: Intervention 10 (Revisit encapsulation with Randomize option)

LESSON 3: PROGRAMMING - Video 6 (Answer all questions on this page) 13-08-2014

1. Look at Joy of Code#6 video. What is Randomization all about? Explain to your friend.
2. State command that makes the turtle turn LEFT?
3. What is the command to make the turtle turn a random number between 0 and 10 to the right?
4. What is the command to turn the turtle either left or right randomly?
5. What is the purpose of the "." or dot in the instruction Greenfoot.getRandomNumber(20)?
6. How can you look at all the classes within the Greenfoot package? Show your friend.

```
public void act()
{
    move(10);
    if (Greenfoot.getRandomNumber(100) > 10)
        turn(10); // turn (Greenfoot.getRandomNumber(40)-20);
//Random number between 0 - 39
    if(getX() < 20 || getX() > getWorld().getWidth() - 20)
        turn(10);
    if(getY() < 20 || getY() > getWorld().getHeight() - 20)
        turn(10);
}
```

6. Let's see where **APOS** fits in.

A = ACTIONS	A command or code to do something e.g. turn(10) or move(10)
P = PROCESS	A sequence or number of steps/code to accomplish something as in act() method
O = "OBJECT"	A name given to describe such a PROCESS e.g. act() or methods called from the act() method.
S = SCHEMA	A number of interrelated PROCESSES to form new SCHEMAS. Can be the World and Actor subclasses which forms a SCENARIO.

Video #6 illustrates the whole Greenfoot framework. Inspect the documentation as explained in the video. How? Click on World and then under Help menu click on Greenfoot documentation!

Thank you. - Hope you enjoyed this. Remember to watch videos 5 and 6! WcR

Appendix M-1: Intervention 11A (Informing the learners of the assessment in a structured manner)

LESSON 4: Preparation for Assessment 20-08-2014

1. **Access site:** <http://moodle.efundo.co.za> **Your login:** c+surname+initial
Password: C1111#c+surname+initial, e.g. Login: *calexanderk* for Kristen Alexander and password will be *C1111#calexanderk*

2. What must I prepare for the assessment?

You will be asked to *practically create a scenario* consisting of a *World* with a background and an *Actor* such as a turtle moving inside the world. The Actor should not fall off, but turn around at the edge of the world. The turtle (actor) should move randomly and turn randomly as it would do in nature.

You will be asked to complete a quiz on the web site above, which you can do from home on Greenfoot logging into the web site. As an example how many classes does one find in Greenfoot? Where can I go to view these classes when in Greenfoot? How can I view the methods of each class within Greenfoot, e.g. In Greenfoot there is a **Greenfoot**-class, which has a method `getRandomNumber()` - How does it work and what does it return? What is an "if" statement and how is it used? How can one determine the x and y coordinate of the turtle? What is the dimension of the world in pixels? Where and how do we determine that?

The **APOS theory**. What does it stand for? What is my understanding of the APOS theory? Can I relate a practical example like "juggling of balls" to APOS. How does Greenfoot relate to APOS? Think about the turtle in the world, providing code (actions - individual commands such as `move(10)`) to execute, which forms a process (sequence of actions to do something), recognized as an object such as the method **edgeOfWorld()** which checks that the turtle does not go beyond the edge of the world, but you can use that method for other actors as well.

3. How should I prepare for the practical?

Download all the videos on system and relook at them from Video #1 through to Video #6. When you do not understand something, watch it over and over again.

After each video take action by doing (executing) the code in Greenfoot or discuss with your parents. View the "Juggling of balls". Think about the APOS theory in acquiring the skill of "Juggling" as you perfect the skill to three balls. Challenge yourself!! In APOS the concept of "Juggling" is seen as an "object". Look at all the Lessons I handed out and redo them. If you lost them, visit the website and download or view them. Remember I asked that you paste them into your **big black book** (BBB). I printed each lesson and handed out to each one in class.

4. Where and when will I be assessed? How long do I have time to prepare for the assessment?

The assessment will be a practical in using Greenfoot at school and a quiz(zes) you need to answer from the comfort of your own home only after preparation. Be prepared/ready from any date starting at end of August to 10 September. I will assess you in small groups at a specific time not to overload the system. Please email me if you have any questions. Enjoy!!

Appendix M-2: Intervention 11B (Assessment in Greenfoot on Encapsulation and problem solving)

Lesson 6

Creating your own background

27-08-2014

1. We will use PAINT to create our own background. Normally the dimensions are 640 x 400, but we can make it larger let's say 900 by 470 cells for cell size of 1 pixel. Open Paint and create a block and colour the inside of the block with light green. Now draw a road on the *canvas* and plant some trees next to the road. You need to drive a car on the road - so do not make it too narrow. Use the eraser to undo changes etc. Add some nice colours into your picture.
2. Once done, *save* it as a jpg file, namely nature.jpg. Ensure you know where you *saved* the file. Create a new subclass (nature) of the World and import the jpg file. Set the dimensions exactly the same as those you used in Paint!
3. Create a subclass for actor and use a car object. When depressing the left and right arrow keys the car needs to *move* ahead and when pressing the up or down arrows the car needs to turn and stay on the road. What method is needed to do just that? How would we reference the method (through which class)? So if no right-arrow key is pressed the car will not *move*. If the left-arrow key is depressed the car will reverse.

Hint: `if (Greenfoot.isKeyDown("left"))`
`move (-5);`

Please note that your assessment is coming up and that the next Video#7 is due. Please log into my site <http://moodle.efundo.co.za>

4. Give examples for the APOS theory in this exercise

A-

P-

O-

S-

Wilhelm Rothman

Lesson 6: Controlling Movement in Greenfoot

2014

Appendix N: Intervention 12 (The Variable in Greenfoot)

LESSON 7: VARIABLE 14-10-2014

1. Variable

A variable may be represented by a symbol such as x, y and z or by a string such as yValue or any string not being a reserved word. It may not contain any spaces or special characters such as "\$, . % ^ &" etc. If you want to represent a space use an underscore e.g. y_Value or x_Value or y-Value. Rather use camel case or Pascal case! A True or a False can also be represented by a type known as a "boolean".

What do you understand by a reserved word and what is camel case or Pascal case?

2. How does one make use of a variable?

It must be declared within the program in between the "class" name and the "act" method and it is normally private for that class. Only the class or instance of that class does have access to that variable.

Example:

- a. Create a "world" and use "Space.jpg" for that and call it "Space".
- b. Create an actor using "rocket.png" and call it "Rocket".
- c. Position the "Rocket" at coordinates (0,200) in the world constructor method. Use the method **addObject(new Rocket(), 0,200);**
- d. Allow the "rocket" to move horizontal by using the variables. What questions must be asked? Which values for x and y axes must change? Now use setLocation(x,y) to move the spacecraft. Remember the act method is called during "run" over and over.
- e. You can also stop the Greenfoot program issuing the command Greenfoot.stop().
- f. Also declare a boolean variable forward and assign the value of "true";.

```
private int xValue = 100;
private int yValue = 200;
private boolean forward = true;
```

3. Move backwards when the rocket reaches the right hand side of the world.

You will have to test if the x-axis value is 580 and then reverse the movement by decrementing the XValue by 5. How do we know when the xValue is 580 - use getX(). The command to decrement the xValue is **xValue = xValue - 5;** or **xValue -= 5;**.

4. APOS.

Think of the actions needed to execute the hence and forth process. There are three important aspects:

- a. Position the space craft at the xValue and yVlue using setLocation(xValue, yValue)
- b. Check or test whether the spacecraft reached the xValue of 580. How can one do that?

Use an "if" statement. **if (getX() == 580) forward = false;** this command is read as: **IF** the value of the object on the x-axis equals 580 or the object reached the value of 580, **THEN** assign false to the boolean variable forward.

- c. Thus the xValue must increment or decrement depending whether the "forward" boolean variable carries a true or a false.

```
if (forward) xValue = xValue + 5;
```

```
if (!forward) xValue = xValue - 5;
```

Please note that the ! sign means NOT. An IF statement must always be true to execute. Think about this: !true is false and !false is true and thus if forward is assigned a false value, then the statement "if (!forward) will be true and the command will/must decrement the xValue to move in opposite direction.

EXERCISE: Create an object and position at bottom in the middle. Now make the object move upwards and downwards.

Challenge for next week: Use a ball and make ball smaller as it move upwards? Hint: Look at the images of the different sizes of balls available. Look at the videos!

Appendix O: Intervention 13 (Moving from Process to Object in APOS using Greenfoot)

Name:

Surname:

10 September 2014

GREENFOOT TEST 2

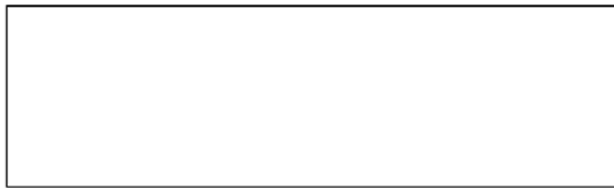
GRADE 8

CURRO PRIVATE SCHOOL

[44 Marks]

Hint: One can start a scenario by using the command `Greenfoot.start()` in your World constructor. This has the same effect when the user clicks on **Run**. So please add that to your scenario. One can also change the **position of an Actor** by using the method `setLocation(x,y)`, if you need to displace the Actor into a vertical direction, other than forward. By keeping the x value constant changing the y value will move the object on the y-axis, like a balloon floating upwards.

1. Create a scenario called **myTest1**, with a background called SandStone. Compile your scenario after each change. (5)
2. Add an Actor (Red Balloon) call it **redBalloon** and another Actor (Blue Balloon) call it **blueBalloon**. Compile again to accept changes. (5)
3. Position the redBalloon at coordinate location (300,380) and the BlueBalloon at coordinate location (0,200), in the World, before/when the scenario starts. First position the balloons in the following drawing, BEFORE CODING. Also indicate the dimensions of the SandStone background here on this rectangle. (10)



4. The red balloon moves to the top and the blue balloon moves from left to right. (10)

State your understanding of actions needed before you do any coding here.

Blue Balloon: _____

Red Balloon: _____

Look at the teacher's scenario and imitate the movement of the two balloons. That is the red balloon moves up to the top of the screen border and the blue balloon moves from left to right across the screen border. First describe the actions needed to accomplish this for the red and the blue balloon.

Appendix P-1: Intervention 14A: Basic creation of scenario with World and Actor classes

TAAK 1 24 FEBRUARIE 2015 NAAM: Luan VAN: de Jager

$\frac{10}{15}$

Doen die volgende taak en beantwoord die vrae OP JOU EIE asseblief!

- Het jy video 3 gekyk? Ja ~~nee~~ Indien nie moet jy dit nou kyk asseblief of kyk of jy die volgende vrae kan beantwoord. Gebruik jou kopfoon vir video.
- Create a world called Desert with actors called Camel and AppleTree. Ensure that you save the scenario as ArabianNights. Add some Camels in the Desert next to several AppleTrees.
- Verduidelik die term scenario. Watter beeld (visualiseer) het/dra jy in jou brein?
Ek sien 'n platform waar ~~daar~~ objekte is wat dalk 'n aksie maak of net stil staan.
- Wat is n "CLASS"? Verduidelik in jou eie woorde hoe/wat jy (dit) in jou gedagte voorstel/sien/visualiseer. Dit is alles wat in jou wêreld kan voorkom. Dit ~~het~~ alles iets met mekaar in gemeen.
- Wat is n "OBJECT"? Verduidelik in jou eie woorde hoe/wat jy (dit) in jou gedagte voorstel/sien/visualiseer. Dit is alles wat ~~op~~ jou wêreld kan voorkom.
- Gee n voorbeeld van n CLASS en gee n voorbeeld van n OBJECT.
CLASS: ~~Dit is die naam~~ Camel Actor ✓
OBJECT: Camel ✓
- Voor jy die ArabianNights scenario geskep het, wat het deur jou brein geflits om dit te kan skep (stappe); dit is, hoe visualiseer (beeld in brein) jy die proses of sien jy dit as n volgorde van aksies hoe om die scenario te voltooi. Beskryf die beeld in jou brein om die scenario te voltooi.
 - Ek het op subclass gegaan en die class ~~die~~ class dessert genoem en 'n sand.png gekies (en dan)
 - Toe het ek by die class Actor 'n objek gemaak deur New subclass te klik en toe noem ek dit Camel en kies 'n Camel.png.

Huiswerk: Kyk asseblief videos 4 en 5 voor jou volgende klas. Skryf in huiswerkboek asseblief.

 - Ek doen toe weer dieselfde en noem dit toe Appte AppleTree en kies toe 'n tree.png en toe compile ek dit. Hoe
 - Toe add ek 'n objek (Camel en AppleTree) ✓

Appendix P-2: Intervention 14B: Manipulation of Actors in a World.

TAAK 2 25 FEBRUARIE 2015

NAAM: Janco

VAN: Greyling

Doen die volgende taak en beantwoord die vrae OP JOU EIE asseblief!

1. Het jy video 4 gekyk? Jaas'jaar Nee. Indien nie moet jy dit nou kyk asseblief of kyk of jy die volgende vrae kan beantwoord. Gebruik jou kopfoon vir video.

2. Voor jy scenario "Moon" skep, wat flits deur jou brein om dit te kan skep (stappe); dit is, hoe visualiseer (beeld in brein) jy die prosedures of sien jy dit as n volgorde van aksies hoe om die scenario te voltooi. Hoe sou jy dit verduidelik aan jou ouer hoe om "Moon" te skep met n "Rocket" actor? Maak Greenfoot oop. Kliek links bo op die scenario knop en kies new. Save jou scenario as in my documents as Moon en sê create.

3. Wat is die doel van die Compile knoppie?

Dit is om jou world (background) op die screen te sit.

4. Plaas n voorkoms van n Rocket in die Moon "World". Wat is die naam van die klas waarvan die "Rocket" objek *metodes* erf? Waar in kode sien jy dit.

inherited from actor

5. Kies nou n metode "move()" en verskaf n waarde aan die metode dat die "Rocket" kan beweeg. Laat hom ook deur n hoek van 90 grade na links draai, dan staan hy regop.

Wys met vinger waar koördinaat (300,100) is. Hoe bepaal jy die koördinaat. *Beskryf.*

Jy kry die koördinaat by inherited from actor en dan int.getX() & get y int.get Y()

Positioneer die "Rocket" by koördinate 300 by 100 en toets jou skatting.

6. Waar sal jy die opdragte move(100) en turn(10) insleutel om die Rocket lewe te gee?

Open editor

7. Verduidelik nou verskil tussen Act en Run knoppies, deur na die Rocket objek te verwys.

Act is stop vir aksie en Run is al die aksies saam

Huiswerk: Kyk asseblief videos ⁵ voor jou volgende klas. Skryf in huiswerkboek asseblief.

Appendix P-3: Intervention 14C: Interaction of Actor within the world solving problems

THE PROBLEM

06-03-2015

Q1: How will turtle be controlled through the use of Java code to turn around when the edge is reached?

Activity 1: Write down in your mother tongue what are your thoughts in preventing the ambulance from falling off any of the four edges of the world.

If the ambulance reaches the right hand side turn 180, else if it reaches the top, turn 180 or if it reaches the bottom, turn 180 or if it reaches the left hand side turn 180

Q2: When confronted with this problem, what should you know as prior knowledge before answering the question?

Activity 2: State all the pre-requisites to answer the question. What should you know within Greenfoot to prevent the ambulance from disappearing off the world?

1. *Dimension of the world*
2. *The methods needed to show the X and Y positions of the object*
3. *The methods to show the height and width of the world – `getWorld.getWidth()` and `getWorld.getHeight()`*
4. *Each statement must be concluded by a semi-colon.*

Q3: What is the control structure to probe if any side was reached?

Activity 3: Rewrite your description of the problem as close in Java code as you can. Consider a condition as closely.

If `getX() > getWorld.getWidth() - 20` turn(15);

Q4: What is your final code structure?

Activity 3: Write the final code step by step and running the scenario to check the outcome.

Appendix P-4: Intervention 14D: APOS theory as tool to investigate problem questions

Van:

Naam:

Groep:

LES 2.3: PROGRAMMERING - Video 4 & 5 (Beantwoord alle vrae op die papier)

26-03-2015

1. Gebruik die scenario soos deur jou in vorige les geskep is.
2. Gebruik n liniaal en potlood en trek die X en Y asse en voltooi die reghoek soos deur die dimensies voorgestel word in die Desert of Sand "World". Verseker dis op skaal, waar elke blokkie 10 pixels voorstel.
 - a. Dui die 0,0 koordinaat aan op grafiekpapier. Hoe kan jy jou antwoord toets in Greenfoot? Beskryf aksies hier.
 - b. Dui die 500, 200 koordinaat aan op grafiekpapier. Hoe sal jy Greenfoot gebruik om te verifieer? Beskryf aksies hier.
 - c. Dui die 600, 400 koordinaat aan op grafiekpapier. Hoe sal jy Greenfoot gebruik om dit te verifieer? Beskryf aksies hier.
3. **Probleem:** Die objek moet weg/omdraai sodra dit die einde van enige van die 4 sye van die wereld bereik. Plaas n objek (voorkoms) by koordinaat 200, 300 wanneer die scenario kompilleer. *Maak die World klas "Desert of Sand" oop en onder die super(600,400,1); voeg instruksie by `addObject(new Camel(), 200, 300)`; en kompilleer waar die Camel jou objek se naam is. Wat verstaan jy onder kompilleer?*
4. As jy kode moet skryf om te bepaal of die objek tot aan die einde van sy wereld beweeg het, kan ons sekere metodes gebruik om dit vir ons te doen. Bestudeer weer die wereld en grafiekpapier en kyk waar die turtle oral van sy wereld kan afval. Skryf neer deur van die Greenfoot metodes gebruik te maak om die volgende te bepaal:
Regterkant van World:

Linkerkant van World:

Bokant van World:

Onderkant van World:
5. Beskryf wat jy verstaan onder n "if" stelling en gebruik dit dan om die objek te laat omdraai as dit die regterkant van die "World" bereik het. Toets jou instruksie in Greenfoot binne die objek se act method.
6. Skryf die kode vir alle sye neer sodat die objek omdraai as dit 40 stappies voor die einde van wereld bereik het. Onthou n kommapunt aan einde van elke stelling! Gebruik HELP funksies!

Appendix P-5: Intervention 14E: The IF statement as a solution to address problems

TOETS 3: Ondersoek die "If Statement" 25 Punte 20-05-2015

Naam: Janco Van: Greyling Groep: 9.1 (Afrikaans)

Skep Toets3 scenario. Nou skep "Resies" world met "Car" actor.
Kompiler scenario.

- Posisioneer die "motor" by koördinate (10,200) in world constructor method. Gebruik die `addObject(new Car(), 10,200)`; metode.
- Laat die "Car" horisontaal beweeg en omdraai aan regterkant en linkerkant.

VRAAG 1: Wanneer moet die motor op horisontaal omdraai? Beskou elke geval. Skryf die kode neer wat jy sal gebruik. Doen elke kant apart. Moenie c doen alvorens jy a en b voltooi het nie.

10 stappe voor hy aan die einde van die kante van die wêreld kom moet hy omdraai (sodat dit nie afval nie). Jy begin deur te se `move(5)` en dan om te verseker dat die car omdraai (aan die regterkant van die X-as, 600) moet jy dan `if (getX() > getWorld().getWidth() - 10) turn(180);` intik.
Om dan te verseker dat die car actor (links van die X-as, 0) omdraai tik jy `if (getX() < 10) turn(180);`

- Jou motor beweeg nou net horisontaal. Jy wil nou die motor in enige rigting laat beweeg en ook wegbeweeg van bo en onderkante van die world.

Beplan eers deur te bepaal wanneer die motor by bo en onderste kante van world kan wees en gebruik die "if" stelling om te verhoed dat die motor links, regs, bo of onder gaan bots, maar wegdraai. Soek metode `getRandomNumber(180)` op in Help se Greenfoot Class Documentation.

VRAAG 2: Wanneer moet motor van bo en onderkante wegdraai? Beskou elke geval apart. Skryf die kode neer wat jy sal gebruik. Doen elke kant apart.

H Dit moet omdraai ^{voor} die einde van die wêreld (a.w. voor dit op die heel (10) punte kom) van die wêreld. Dus het ek gesê dat die actor omdraai 10 stappe voor dit aan die einde van die wêreld kom. Ek het dit ook laat beweeg met 5. Ek het dan die `getRandomNumber` metode gebruik om die objekt(actor) enige grade tussen 0 en 180 te laat draai. Ek het dit ook so ingetik dat hy horisby die X en Y-asse omdraai.

- Skep n paar bome en voeg by op lukraakte plekke. Verseker die bome verskyn wanneer jy kompileer. Skryf kode sodat die motor wegdraai sodra dit met bome kontak maak. Wenk: Soek onder Help-Greenfoot Class Documentation, die metodes van die Greenfoot klas. (5)

Appendix Q: Rollout to WCED Schools



Appendix R-1: Task for interview 1A (Algebra Exercise on Simplification)

NAAM:

VAN:

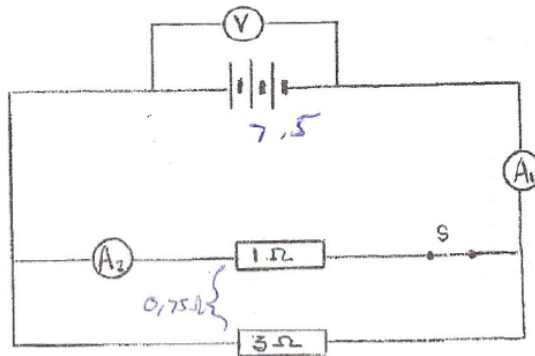
GRAAD 9:

VEREENVOUDIG OF FAKTORISEER:

1. $(-3x^2y^3)^3$
2. $(2x + 1)(3x - 2) - (x - 3)(x + 5)$
3. $(b + a)^2 - (b + a)$
4. $2a^2 - 22a - 24$
5. $(3x - 2y)(3x + 2y)$
6. $(2a - 3b)^2$
7. $18x^2 - 2$
8. $(2x - 3)(4x^2 - 5x + 1)$

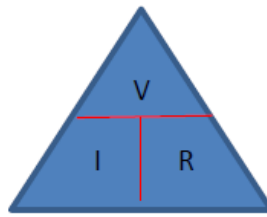
Appendix R-2: Task for interview 1B (Science Assessment Question)

BESTUDEER DIE VOLGENDE STROOMBAAN



9. Elke sel het n PV van 2.5 V. Wat is die lesing op die volmeter?
10. Bereken die totale weerstand van die resistors in parallel
11. Bereken die lesing op die ammeter A1?
12. Wat is die lesing op die ammeter A2?
13. As S gesluit is wat is die lesing op ammeter vir A1?

Appendix R-3: Interview: Voltage-Ampere-Resistance pyramid



Appendix R-4: Interviews on Mathematics and Science

Interview 1: MDL

Me: Do you take extra maths classes?

MDL: Now and then. I am carrying on with normal mathematics.

Me: Do you use a calculator?

MDL: Yes I must use a calculator.

Me: I explained the power of exponents in multiplication and not to the power of

MDL: I cannot calculate the power of exponents

Me: You struggle because you do not know your rules? (I tried to explain the basics by guiding the learner.)

MDL: I am very nervous in attempting no 5. $(3x - 2y)(3x + 2y)$. Have not done this in a while and I am not sure how to simplify the expression.

Me: I tried to guide her in the right direction. I explained to use substitution where $a=3x$ and $b=2y$ so the expression changes to $(a - b)(a + b)$.

MDL: Oh ok now I see it is $a^2 - b^2 - ab + ab = a^2 - b^2$, so it is $9x^2 - 4y^2$

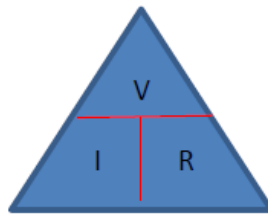
Me: Do you see the difference between two squares

MDL: No I cannot remember, but oh yes now I remember.

Me: Let's talk about the circuit diagram. What is V?

MDL: I need the diagram to infer the answers $V=IR$.

Me: Learner now could easily use the triangle to infer the answers, but no real understanding of the real world problem. The learner has no connection with the circuit and its properties other than applying the diagram given in the triangle.



Me: What are the formulae for circumference and area?

MDL: I struggle with determining which is which.

Me: Think about square as to power of 2. So the circumference is $2 \times \pi \times \text{Radius}$ and area is $\pi \times \text{Radius}^2$.

MDL: Oh yes I now understand how to memorise and know the difference.

Me: What is area of rectangle and triangle and so on?

MDL: No I do not know the formulae. I will have to go home and memorise them. Hhm, (quiet)

Me: What must you do for tomorrow's test?

MDL: I need to memorise the formulae and associate them with a specific Figure.

Interview 2: MA

I interviewed learner MA and we spoke about their test to be taken down the next day. Upon asking the learner what are the sum of the angles of a rectangle, he stated 45 degrees. I asked him to draw a rectangle. After drawing the rectangle the learner could easily calculate the answer for he took a physical action and represented the Figure on paper and could reason about the figure by applying other schemas of a triangle to it. Another problem was that the learner did not know the formulae associated with a geometrical figure such as circle, triangle or rectangle, which comes down to memorising these formulae.

Me: Write down what problems you have in maths. What do you find difficult?

MA: Must I say what my problem is?

Me: Yes.

MA: Problem with memorising. I have problems with exponents.

Me: What do you do about your problems?

MA: I knew my rules but just forgot them. I am unsure about them. I take extra Master Maths classes. I also have problems with geometry. I simply do not know my rules.

Me: OK if I ask what is area of trapezium. Would you know what to do?

MA: Is it not length times hhm... hhm

Me: What is area of triangle?

MA: Length x Base. When I study for test I forget the stuff I learnt.

Me: So what you say you have a problem with memorising. So what do you do about it, how do you counter this problem?

MA: My parents helped me initially but they cannot anymore.

Me; Does Master Maths help?

MA: It increased my mark to 60%.

Me: Do you have problem with lines – Understanding?

MA: Yes lines are problems. I get confused with calculations like factorisation. Data handling is work I need to study. If I prepare for it I am fine. The formulae of different figures are a problem.

ME: Why is that figure a trapezium. What distinguishes the figure from a triangle?

MA: Not sure.

ME: Let's look at the problem of area. Write down what you know about area and we can investigate if "what you know" can help you to solve other area problems. You know a triangle area formula. Look at what you can infer from the figure.

MA: I now see that I actually knew how to calculate the area without knowing the actual formulae, but by breaking the figure into two triangles, I can calculate $2(\frac{1}{2} \times \text{Base} \times \text{perpendicular height}) = 1 \times \text{Base} \times \text{perpendicular height}$. So can use other things I know to solve the problem.

ME: So to know the formula is a shortcut. When you struggle do not look at the problem, but what the problem provides me to solve it. So what is the formula of the trapezium?

MA: It is $2(\frac{1}{2} \times \text{Base} \times \text{perpendicular height}) = 1 \times \text{Base} \times \text{perpendicular height}$. So I can infer the area without it being memorised. I see now I am not hopeless.

ME: Yes you possess the answer. What is square's area?

MA: side x side x side?

ME: No just side x side. Please draw it and think again.

MA: Oh I now see it's only side x side.

ME: You took an action by drawing the figure which clarified your understanding. The radius of a circle is 6 cm, what is the circumference?

MA: hhm... hmmm.

ME: OK the area of a circle is what?

MA: it is pi x radius squared?

ME: OK how can you distinguish area from circumference? Take the squared sign and put in front as $2 \times \pi \times \text{radius}$ and not $\pi \times \text{radius squared}$.

MA: Oh now I understand the difference.

So the learner moved to the process phase without an action phase. The process stage was thus incorrect, due to a lack in the action phase. The enactment or visualisation was absent. The learner thus haphazardly guessed his version of a solution to calculating the area of a triangle.

ME: Do the first question. $(-3x^2y^3)^3$

MA: OK it is the x exponent x 3 and the y exponent x 3 and the 3×3

ME: Remember your exponent laws.

MA: OK it is $3 \times 3 \times 3 = 9$

ME: One fault you made is the sign which you did not consider but just carried the minus over. Remember $-x \cdot - = +x$ $- \cdot - = +$ So your sign is determined by the even or odd number of times multiplied.

Algebraic expression was also calculated as a process without physically writing it down in taking an action. The learner also just calculated squares and multiplication without considering the sign of the number. The important aspect is taking an action by writing the values down and not just calculating values. Even where an expression consists of many x and y's the learner could not use abstraction by assigning an alphabetic letter to a sequence of numbers.

After the learner came forward by taking an action and remembering the rules where applicable, he wanted to do the last problem and enjoyed the challenge. He soon bought into taking action when he was unsure about a memorised process.

Interview 3: KW

Me: Do you want to pursue normal mathematics this next year?

KW: I want to but maths percentage is 65%

Me: What problems do you have in mathematics you think and do you take extra classes?

KW: I will write down. I take maths at teacher at school, but I want to change for she thinks the same line of thought. I have problems with lines and tomorrow I am writing geometry tomorrow.

Me: So do you have problems with geometry?

KW: I think I have problems with something I learnt yesterday. Translations and rotations are problem sometimes.

Me: What is the problem?

KW: The x and y's can be confusing when I rotate?

Me: Think about rotation of the following? 270 degrees anti clockwise is the same as 90 degrees clockwise?

KW: Oh yes I see. Maybe the rules for geometry tomorrow are maybe difficult. The names of the figures are sometimes difficult. Area is difficult for me.

Me: Let's look at an example.

KW: If I know the rules it might be easier.

Me: Identify the figure and check what they ask. You must know the formula and you must know those rules before you enter test tomorrow. Do you agree it's your own fault if you do not memorise the figures and rules.

KW: Yes I agree, I must spend time on the figures and formulae.

Me: What is area of triangle?

KW: half base time perpendicular height (Me correct and help her to get to correct answer)

Me: What figure is the following?

KW: It is a rectangle.

Me: What are the properties of a rectangle? Is it a rectangle?

KW: The rectangle is actually skew and hence a parallelogram.

Me: You must know the properties. Do you see? How will we calculate the area?

KW: Length x breadth x height?

Me: Let's look at the figure. Which figure's area do you know?

KW: I know the area of a triangle.

Me: OK draw the parallelogram and inspect the structure. Look at the figure and what do you see?

KW: It is two triangles.

ME: It is correct and what is area of ONE triangle in that figure?

KW: Can I calculate the area for the triangle is not in the middle regarding the height?

Me: Any triangle has a fixed formula which is $\frac{1}{2} \times \text{Base} \times \text{perpendicular height}$.

So how many triangles do we have? You can calculate that in your head, bit write it down.

KW: So it is $\frac{1}{2} \times \text{Base} \times \text{perpendicular height}$. But I have 2 of these triangle so the formula now is: $2(\frac{1}{2} \times \text{Base} \times \text{perpendicular height}) = 1 \times \text{Base} \times \text{perpendicular height}$.

Me: So does the rectangle formula of Base x Height make sense now?

KW: Yes I now understand where the formula comes from.

Me: Solve the first simplification namely $(-3x^2y^3)^3$

KW: The factorisation and simplification is a very difficult thing for me.

Me: Her answer was 9, but after reconsideration she said 27. Only when she wrote down the expression, she realised that the minus will remain. Asking her what the problem was she stated that the brackets cause confusion. Her problem was the 9, instead as 27. She could not calculate the answer and never transferred her finger counting to a process. She then mentioned APOS in actions and processes. She then admitted that using her fingers creates a problem.

What was your problem to calculate 3 to the power of 3.

KW: I could never calculate any such values without using my fingers.

Me: Remember APOS? You still count on your fingers which YOU must change to internalise those finger counting.

KW: I sometimes not count using my fingers but I think in my mind using my fingers to count.

Me: No 7. $18x^2 - 2$. I lead the learner with questions to think about the common factor which is 2. She then realised that $2 \times 9 = 18$, but must do this for the other term as well. She could not recognise the difference between squares of two numbers. I used another easier example to explain the difference of two squares. The learner then stated that she must know her rules which were lacking and memorising certain actions. The learner admitted APOS which she learnt in programming and applied that in mathematics.

Me: You identify 9 as common factor and remove the 9. What do you have left?

KW: I have 0 left.

Me: No, think again.

KW: Oh ok I have 9 left

Me: Have you heard of difference of two squares?

KW: No

She then asked about the gradient of a line, which indicated she started to enjoy the interview and the mathematical reasoning. I suppose it boiled down to understanding her and overcoming her fears.

The interview on the electrical circuit was much clearer to identify the problem. The learners were taught at all schools within my interviews to draw a triangle with VIR. Rather alarming that a concept image is also a practical plan initiated by many schools to focus on better marks than better understanding and not enabling the learners to grasp the concept as per definition.

The next aspect was about abstraction and that the learner can experience why abstraction plays an important role in their conceptual understanding of any concept. The research became a bit fuzzy in that I talk and assess certain important concepts with the learners, but there is nothing tangible as a footprint of my research or any tangible ideas which the learners may take with on their journey to tertiary education. The quantitative research which to establish reason for failure and non-interest in subjects like mathematics and science did not appeal to me as the route to take and I had to determine the status quo of what I found in the schools.

Appendix S: Greenfoot Name Badge



Appendix T: Order Form for the Greenfoot Badges



TEL: +27 21 510-1810 – FAX: +27 21 510-1812
E-mail: orders@namebadges.co.za

ATT: Wilhelm Rothman	CO: University of Kent
TEL: 021 913 4083	Fax: 021 591 3087
DATE: 19 May 2015	Email: rothmanw@Telkomsa.net

APPROVAL OF ARTWORK

Below please find layout and artwork for your Badge.

Background Color: White Background

Size: 25x34mm Oval

Printing: Digital Vinyl

Finishes: Epoxy

Fittings: Clutch Pin + Brass Metal Clutch

Quantity: 100

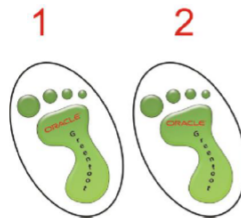
Price per Badge: @ R25.00 excl vat each

Artwork and Setup @ R200.00 excl vat

Delivery Cost: @ R65.00 excl vat

Payment Terms: Full Payment is required upfront before we commence with manufacturing.

Option



Approved: (Name)

..... (Date)

..... (Signature)

Please note:

For manufacturing to commence the artwork approval should be signed off by:

- Scanning signed document and emailing back to sender or;
- Typing in the appropriate spaces name, date, and signature and emailing back to sender.
- If faxed please mark clearly if there are any changes as faxes tend to come out dark and unreadable.

Appendix U: First Game Development

LESSON 8:

MY FIRST GAME

17-10-2014

1. BRIEF

You are tasked to write code for a game called Ping-Pong. This is one of the first games played in early computing. Remember APOS? We now need to investigate the components of this game and you and your partner will develop the game in a modular way, finding all the actions needed.

OUTCOME: Create the Ball, the Paddle, the Green background classes and during compilation position the paddle in the middle of the Green world slightly above the bottom, but paddle aligned in the world and ball aligned on the paddle, i.e. centered

2. COMPONENTS OF THE GAME

The game consists of a world (upright), a paddle or batten AND a ball that bounces off the paddle and off the walls of the world. When this game is fully functional the ball will bounce at angles and the player must move the paddle to prevent the ball from bypassing the paddle. If this happens it's game-over. Otherwise the ball will collide and bounce off the paddle into area above the paddle.

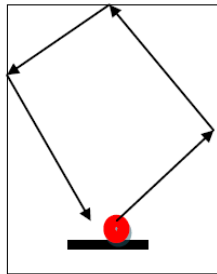


Figure 1: Representation of the ball on the paddle and a possible bounce path

The user must with the aid of the right and left arrow keys move the paddle on the same horizontal (x-axis). When the game starts the ball will be positioned exactly in the middle of the paddle and when the user presses the spacebar the ball will move upwards at an angle. The user must now move the paddle to catch the ball and force it to bounce back.

3. GIVEN

Find the image for the background, the image for the paddle and the image for the ball on the website under Activity 8 or from your local server or create them using Paint. Discuss the game with your partner and note the following points:

What objects are needed?

Upright world with dimensions of probably 460 by 520 or super(460, 520, 1). You will notice that this implies a total x-value of only 460 but the height or y-value is 520 and hence Figure 1's upright rectangle.

So let's create these objects. Create your world and the paddle and ball classes. Import these images as needed. Now place the images as in Figure 2 at the desired position.

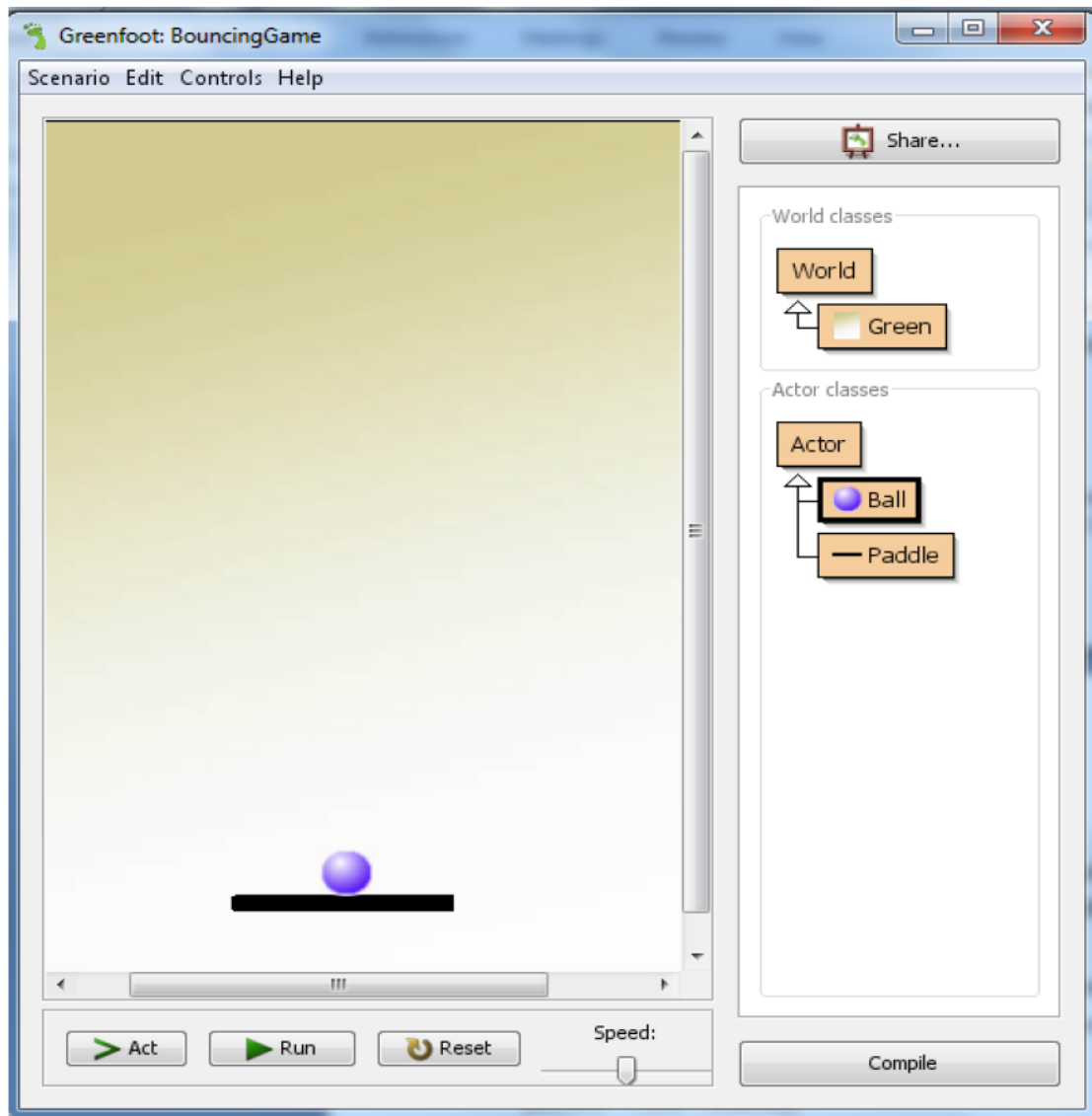


Figure 2: Greenfoot scenario after compilation

The problem with this is that you placed the paddle and ball at those specific locations. In a nutshell the paddle must be positioned or aligned at the centre of the width of the background called Green and slightly above the bottom. The ball must be positioned or aligned in the centre of the paddle in turn.

Thus when the game starts up, the ball and the paddle must be positioned correctly within the code. How would we do that? Well we need to ask the right questions!

Do we have a method that can position the paddle image at the desired location? Well we can inspect the location where we placed the paddle, but it might not be 100 % accurate. So what then?

Make use of the methods within the world class. When within a class such as the actor one can use those methods by name if they belong to the actor class, e.g. we create an instance of Paddle called paddle and then determine the location of the Paddle instance by issuing the command `paddle.getX()` and `paddle.getY()`. When within another class one needs to prefix the method with the name of the class, one wants to perform some method. In the world class declare a variable namely :

```
private Paddle paddle;
private Ball myBall;
//Let's create an instance of the Paddle class in the world class
    paddle = new Paddle();
//Position the paddle object at the location, which is the width of the world divided by 2, and 40
pixels from the bottom of the screen, i.e
getWidth is 460 / 2 = 230 and the getHeight() = 520 - 40 = 480
    addObject ( paddle, getWidth() / 2, getHeight() - 40);
    addObject (myBall, paddle.getX(), paddle.getY()-18);
```

Your World class code should now look like this:

```
public class Green extends World
{
    /**
     * Constructor for objects of class Green.
     */
    private Paddle paddle; //Reference to Paddle class
    private Ball myBall; //Reference to Ball class

    public Green()
    {
        // Create a new world with 460x520 cells with a cell size of 1x1 pixels.
        super(460, 520, 1);
        paddle = new Paddle();
        myBall = new Ball();
        addObject (paddle, getWidth() / 2, getHeight() - 40);
        addObject (myBall, paddle.getX(), paddle.getY()-18);
    }
}
```

Explain the implication of the code above, namely why the paddle will be centered and why the ball will be centered within the paddle. Find more interesting methods by inspecting the methods of the World and Actor classes.

Next time we will release (Process 1) the ball and move the paddle with the arrow keys left (Process 2) or right (Process 3). In the meantime describe all actions to take care of these processes, before coming to the next class.

Appendix V: Putting it Together

LESSON 9 1 HOUR

Putting it together:

Please find the image folder and sounds folder with all your goodies. Let's analyse the "problem":

You are asked to create a piano with 12 – white keys. You also need 8 – black keys in between the white notes. The piano keys rest onto a wooden surface.

1. Make a drawing of such a keyboard – Textual Storyboard



2. Discuss with your neighbor how we will put this "Game" together.

You have one white and one black key with their alternate keys with a slight colour change when depressed.

Phase 1 What do we have?

- You have 12 W + 8 B keyboard characters to be stored somewhere.
- *For each key*
- You have 12 white key sounds.
- You have 8 black key sounds

We can describe our world as the **Piano** world

We can describe our Actor as the **Key** class of which we will have 12 white keys and 8 black keys

We can either create a variable for each key and each sound OR we can introduce an array to store all whiteKeys and another for all blackKeys

We can also create parallel arrays to store all whiteNotes and all blackNotes.

Phase 2 The design

Open a new scenario and call it Pianissimo. After saving the scenario, copy the images and sounds folders into the scenario.

Add the Piano world and the Key Actor as stated above and set their images to "wood" and "white" respectively. Ensure your Piano world is: super(800, 340, 1);

Now test your world by creating instances of the Actor and put the key(s) onto the wooden surface.

Think about what the "white" key must do in its act-method. You need to press the white-key and when depressed play the corresponding sound. At the same time when the key is depressed the colour must change as in the Crab-example.

```
private Boolean isDown;

public void act()
{
    if (!isDown && Greenfoot.isKeyDown("g")) {
        play();
        setImage("white-key-down.png");
        isDown = true;
    }

    if (isDown && !Greenfoot.isKeyDown("g")) {
        setImage("white-key.png");
        isDown = false;
    }
}

Public void play()
{
    Greenfoot.playsound("3a.wav");
}
```

Abstraction needed for Key in order to make each one unique

Create a constructor for each key with its own unique set of properties. Ensure the sounds and keys are in order (parallel)

Motivations

Using a more guided approach in Greenfoot:

A less-skilled learner will rather opt for a less-guided approach for according to Clark, Kirschner and Sweller (2012) a more guided approach requires learners to provide a more attention-driven approach. On the other hand more-skilled learners will opt for a guide. It is argued that worked examples are important to minimise time spent on concepts. A worked example (Clark et al., 2010, 2012; McPhail, 2016) usually originates.

Appendix W: Table of interventions and Actions

Table 3.3 Interventions and sub-interventions in this research

DCA: Data Collection Action taken by the researcher.

Observation: O, Lectures: L, Interview: I, Reflection R, Practical P

No	Activity	Description	DCA
1	Intervention 1A	Abstraction (Abstract Thinking) Assessment	O
2	Intervention 2	Implement Greenfoot (Circumscribed)	P
3	Intervention 2A	An enhanced Theoretical Conceptual Framework	R
4	Intervention 2B	Introduction of a genetic decomposition (GD)	L,P
5	Intervention 2C	Introduction of an enhanced GD	L,P
6	Intervention 3	Interaction with the Moodle LMS(Circumscribed)	L,P
7	Intervention 3A	Tools for Moodle LMS	P
8	Intervention 3B	Juggling enactment for Moodle	O,L,P
9	Intervention 3C	Moodle and Generalised Terminology	L,P
10	Intervention 4	Creating a Moodle Learner Management System (LMS) (Circumscribed)	P
11	Intervention 4A	Creating a Linux Server with external access	P
12	Intervention 4B	Creating a Cloud-based Moodle LMS	P
13	Intervention 5	Greenfoot Access (Circumscribed)	L,P
14	Intervention 5A	Introduction to Greenfoot	L,P
15	Intervention 5B	Revisit previous Activities	L,P,O
16	Intervention 6	Applying Process and Object within mathematics	L,P
17	Intervention 7	Greenfoot as Process and Object	L,P
18	Intervention 8	Rollout of code in Greenfoot	L,P,O
19	Intervention 9	Making decisions towards Encapsulation	L,P,O
20	Intervention 10	Revisit encapsulation with Randomize option	L,P,O
21	Intervention 11	Assessment (Circumscribed)	P
22	Intervention 11A	Informing the Learners of the Assessment	L,O
23	Intervention 11B	Assessment in Greenfoot on Encapsulation and problem solving	L,P
24	Intervention 12	The Variable in Greenfoot	L,P
25	Intervention 13	Moving from Process to Object in APOS using Greenfoot	L,P
26	Intervention 14	GD creation on IF statement in Greenfoot (Circumscribed)	L,P
27	Intervention 14A	Basic creation of scenario with World and Actor classes	L,P,O
28	Intervention 14B	Manipulation of Actors in a World	L,P,O

No	Activity	Description	DCA
29	Intervention 14C	Interaction of Actor within the world solving problems	L,P,O
30	Intervention 14D	Adding graph paper as part of GD to develop algorithm	L,P,O
31	Intervention 14E	The IF statement as a solution to address problems	L,P,O
32	Intervention 15	Testing Greenfoot to be accepted among teachers	L,P,O
33	Intervention 16	Creating an Arcade Game	L,P,O
34	Intervention 17	Manufacturing Greenfoot Badges	P

Appendix X: FEDS for Artefact Evaluation

Structure of the Framework for Evaluation in DS Research (FEDS) (Venable, Pries-Heje & Baskerville, 2016:78-82)

Option: Used(U) Ignored (I)

No	Activity	Description	Option
1	Formative evaluation	Empirically based interpretations that provide basis for successful action to improve characteristics or performance of the evaluand	U
2	Summative evaluation	Empirically based interpretations that provide basis for shared meanings about evaluand in different contexts – measure results of completed development	U
3	Ex Ante evaluation	Evaluates candidate systems before rollout	U
4	Ex Post evaluation	Evaluates systems after rollout.	U
5	Functional purpose of evaluation	Formative and/or summative	U
6	Paradigm of the evaluation study	Strategies based on i) naturalistic ii) artificial	U
7	Strategies for evaluation	<ul style="list-style-type: none"> i) Quick & Simple artefact ii) Human Risk & Effectiveness artefact iii) Technical Risk & Efficacy artefact iv) Purely Technical artefact 	<ul style="list-style-type: none"> I U U I

Appendix Y: Themes

Find#	Summary Findings	Category	Themes
14D-2	Learners went back to action to understand problem	Action	APOS
14B-3	Learners discovered how to debug step-by-step	Action	
14A-2	Learners went back to action to understand problem	Action	
14B-1	Learners re-enforced understanding through action	Actions	
16-1	Learners illustrated APOS qualities	APOS	
3B-3	Learners illustrated APOS qualities	Process	
5B-3	Learners schema was better developed	Schema	
8-3	Learners schema was better defined in their minds	Schema	
2-4	Learners schema for maths had to be rethought	Schema	
13-1	Learners enhance understanding in Visual programming language	Schema	
13-2	Learners schema became important	Schema	
14A-3	Learners schema became important	Schema	
14A-4	Learners schema became important	Schema	
14C-1	Learners had urge to move through schema stages	Schema	
14E-1	Learners schemas played prominent role in their understanding	Schema	
14D-2	Learners schema expanded	Schema	
14B-3	Learners schema expanded	Thoughts	
15-5	Beliefs created challenges	Beliefs	
15-6	Beliefs helps with learning	Beliefs	
15-4	Teachers made linkages with “met-befores”	Met-befores	
3C-6	Negativity towards programming language due to unknown as in maths case	Met-befores	
2A-2	Learners fixate on concept images	Pop-Ed	
2B-3	Learners are not keen to to explore	Cognitive Load	COGNITIVE BALANCE
2C-1	Learners welcome LMS as resource	Cognitive Load	
2C-2	Learners welcome LMS as resource	Cognitive Load	
3A-2	Learners had too much to memorise	Cognitive Load	
3B-1	Interest in a topic generates positive attitudes	Cognitive Load	
3B-2	Interest in a topic generates positive attitudes	Cognitive Load	
2-1	Learners are not keen to to explore	Cognitive Load	
2-2	Learners are not keen to to explore	Cognitive Load	
14C-2	Learners used coding to describe algorithm	Cognitive Load	
14E-2	Learners used coding to describe algorithm	Cognitive Load	
2A-1	Learners want to explore work they understand	Met-befores	
1-1	Learners used enactment to avoid abstraction	Abstraction	COMPUTATIONAL THINKING
1-2	Abstraction lacks from learners performing mathematics	Abstraction	
3C-4	Learners show lack of knowledge and skills	Abstraction	
1-3	Learners show lack of knowledge and skills	Abstraction	
6-1	Teachers found topic challenging	Abstraction	
9-1	Learners applied abstraction through encapsulation	Abstraction	
9-2	Learners enhance understanding in Visual programming language	Abstraction	
10-1	Learners used built in method to solve problem	Abstraction	
10-4	Learners used built in method to solve problem	Abstraction	

Find#	Summary Findings	Category	Themes	
11B-1	Learner linked programming language and Windows Tools	Abstraction		
11B-2	Learners applied abstraction through encapsulation	Abstraction		
14D-3	Learners used built in method to solve problem	Abstraction		
6-2	Learners followed APOS	Process		
8-1	Learners followed APOS	Process		
3C-1	Learner links absent between mathematics in Greenfoot	Relation		
8-4	Learner linked mathematics in Greenfoot	Relationships		
14A-1	Learners revisited GD	Actions		LEARNING
15-3	Teachers had similar challenges than learners	Met-befores		
15-2	Teachers made linkages with "met-befores"	Relationships		
3B-4	Learners fixate on concept images	Skill		
5A-2	Technical challenges influenced learning	Teaching		
5A-3	Technical challenges influenced learning	Teaching		
5A-4	Technical challenges influenced learning	Teaching		
11A-1	Learners academic world must be structured.	Teaching		
12-4	Learners academic world must be structured and official.	Teaching		
15-1	Teachers found topic challenging	Teaching and Learning		
3A-3	Learners could work on their own if they are given guidelines	Teaching and Learning		
3C-5	Learners show lack of knowledge and skills	Teaching and Learning		
2B-1	Learners are not keen to to explore	Moodle	LMS	
2B-2	Learners are not keen to to explore	Moodle		
3A-1	Learners liked a change in behaviour	Moodle		
4B-1	Moodle solved challenges	Moodle		
4B-2	Moodle solved challenges	Moodle		
4B-3	Moodle has costs	Moodle		
4B-4	Moodle solved challenges	Moodle		
5B-1	Moodle solved challenges	Moodle		
5B-2	Moodle solved challenges	Moodle		
11A-2	Learners used Moodle for preparation	Moodle		
3B-5	Moodle solved challenges	Moodle		
16-2	Learners coding enhanced	Coding	PROGRAMMING LANGUAGE	
16-3	Learners coding time intensive	Coding		
3C-2	Learners lack programming language knowledge	Coding		
3C-3	Learners lack programming language knowledge	Coding		
5A-1	Learners enhance understanding in Visual programming language	Coding		
7-1	Learners enhance understanding in Visual programming language	Coding		
7-2	Learners enhance understanding in Visual programming language	Coding		
8-2	Learners enhance understanding in Visual programming language	Coding		
8-5	Learners enhance understanding in Visual programming language	Coding		
9-3	Learners see value of control structures	Coding		
9-4	Learners had challenges to understand execution of Greenfoot	Coding		

Find#	Summary Findings	Category	Themes	
9-5	Learners enhance understanding in Visual programming language	Coding		
2-3	Learners show challenges with IDE of programming language	Coding		
10-2	Learners enhance understanding in Visual programming language	Coding		
10-3	Learners enhance understanding in Visual programming language	Coding		
11B-3	Learners enhance understanding in programming language	Coding		
12-1	Learners enhance understanding in programming language	Coding		
12-2	Learners enhance understanding in programming language	Coding		
12-3	Learners enhance understanding in programming language	Coding		
12-5	Learners found syntax challenging in coding	Coding		
13-3	Learners enhance understanding in programming language	Coding		
14B-2	Learners enhance understanding in programming language	Coding		
4A-3	Technical networking allow learner external access	Networking		TC
4A-1	Technical wizardry can save costs	Technical		
4A-2	Technical logic can secure productivity	Technical		
4A-4	Power failures high jack technical expertise	Technical		