Identifying threshold concepts and misconceptions in Computer Programming for 14-17-year-old students by exploring teachers’ perspectives and students’ affective dimensions of learning

Kallia, Maria

Awarding institution: King's College London

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without proper acknowledgement.

END USER LICENCE AGREEMENT

Unless another licence is stated on the immediately following page this work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International licence. https://creativecommons.org/licenses/by-nc-nd/4.0/

You are free to copy, distribute and transmit the work

Under the following conditions:

• Attribution: You must attribute the work in the manner specified by the author (but not in any way that suggests that they endorse you or your use of the work).
• Non Commercial: You may not use this work for commercial purposes.
• No Derivative Works - You may not alter, transform, or build upon this work.

Any of these conditions can be waived if you receive permission from the author. Your fair dealings and other rights are in no way affected by the above.

Take down policy

If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Identifying threshold concepts and misconceptions in Computer Programming for 14-17-year-old students by exploring teachers’ perspectives and students’ affective dimensions of learning

Maria Kallia

Thesis submitted in fulfilment of the requirements of a degree of Doctor of Philosophy, King’s College London, April 2020
Abstract

Threshold concepts, as introduced by Meyer and Land (2003), have been the focus of many studies which aim to explore and understand students’ conceptual difficulties within the disciplines. In computer programming specifically, these efforts focus on higher education and concentrate mostly on the broad area of programming. Now that programming is increasingly being embedded in schools, it is critical for research to identify and consider the difficulties that school students confront in this subject. To this end, this study aims to address a gap in our understanding of concepts and skills involved in the practice of computer programming in school settings and specifically, at key stage 4 and 5 (14-18-year-old students).

The thesis explores students’ difficulties in programming through the three lenses of threshold concepts, misconceptions and affect, and narrows down the problem area to a specific topic in programming, that of functions. Research in this area of programming highlights that functions are problematic for students, an area where misconceptions arise, and a potential threshold concept itself. Therefore, the thesis first identifies potential threshold concepts and misconceptions that students experience in functions by exploring both computing teachers’ perspectives as well as students. To explore in more detail how students experience liminality, a particularly unstable and difficult state, the thesis investigates the role of the affective domain and particularly the role of several affective constructs on students’ progress in this subject: self-efficacy and calibration, task value, sense of belonging, motivation, and computer science identity.
The results of this investigation led to the characterisation of parameters, parameter passing and return values as a threshold conception and of procedural decomposition as a procedural threshold (threshold skill) in computer programming. Moreover, the study revealed three new misconceptions that students experience in functions as well as that students in liminal space experience statistically significant lower levels of self-efficacy in programming and computer science identity than students in post-liminal space. The results also highlight the role of troublesome knowledge on students’ self-efficacy, motivation and sense of belonging and lead to a discussion regarding the limits between liminality and post-liminal state.

The findings of the research study contribute to the methodology of identifying threshold concepts, to the threshold concept framework and to computer programming education.
Acknowledgements

Pursuing a PhD was a life’s aim for me, and it came true due to Dr Sentance, who trusted me to do a PhD with her, and my husband’s support. I will always be grateful to both who made my dream come true.

I was so lucky and honoured to share this journey with both my supervisors, Dr. Sentance and Prof. Healy. With your invaluable support, expertise, patience and tolerance, you accompanied and guided me through this journey. Thank you so much for being there, listening to all my concerns and always motivating and encouraging me to keep going. I am grateful for all the time you dedicated to me and most importantly, for helping me overcome my limits.

Besides my supervisors, I would like to thank my husband Marinos. He was the one who convinced me to start this PhD and the one that supported me the most with all the difficulties. Most of all, I would like to thank him for his patience and understanding for all the days I was not there and his continuous belief on me.

My sincere thanks to my sweet sister Helen, who helped me with this research and listened to all my problems and complains. To my best friends George and Valia, for just being there and for all the drink-nights after a bad day. To my beloved parents, Maria and Pantelis, who taught me to follow my dreams no matter the difficulties and were always with me even though they are far. Finally, to my precious niece Maria, who still wonders why I left Greece and haven’t returned back, thank you for trying to understand, for being proud of me, and for all our skype talks that made me laugh and forget.
# Contents

Abstract .................................................................................................................................................. 2
Acknowledgements ................................................................................................................................. 4

1 Chapter 1: Introduction ......................................................................................................................... 12
   1.1 Background to my research - Curriculum transformation ....................................................... 14
   1.2 My motivation in this research .................................................................................................. 16
   1.3 Research Problem and Context ................................................................................................. 18
      1.3.1 Research problem ............................................................................................................... 18
      1.3.2 Research Statement, Questions and Methodology .......................................................... 20
         1.3.2.1 Phase 1: Identification of potential threshold concepts: a Delphi study .................... 21
         1.3.2.2 Phase 2: Teachers’ experiences with potential threshold concepts: An interpretative phenomenological analysis ................................................................. 22
         1.3.2.3 Phase 3: Affective dimensions of learning, misconceptions and threshold concepts ..... 23
   1.4 Overview of the thesis .................................................................................................................. 24

2 Chapter 2: Literature review .............................................................................................................. 27
   2.1 Threshold concepts ..................................................................................................................... 27
      2.1.1 Threshold Concepts and Liminality .................................................................................... 27
      2.1.2 Characteristics of Threshold Concepts ............................................................................. 30
      2.1.3 Threshold conceptions and Procedural thresholds ............................................................ 33
      2.1.4 Threshold concepts and pedagogical implications ............................................................. 36
      2.1.5 Methodologies of Identifying Threshold Concepts in disciplines and Critique ............... 39
         2.1.5.1 Interviews with Students and Teacher/Lecturers ....................................................... 41
         2.1.5.2 Tests, Quizzes and on Field Observations ................................................................. 45
         2.1.5.3 Phenomenography and Reflective essays ................................................................. 47
      2.1.6 Threshold Concepts in Computer Programming ................................................................. 49
      2.1.7 Concluding Remarks and Critique ...................................................................................... 57
   2.2 Misconceptions ............................................................................................................................. 59
      2.2.1 Misconceptions in Learning ............................................................................................... 59
      2.2.2 Misconceptions in Computer Programming ....................................................................... 62
2.2.2.1 Fragile Knowledge and Sources of Misconceptions in Computer Programming .......................................................... 63
2.2.2.2 Examples of Misconceptions in Programming .......................... 68
  Variables and Assignment........................................................................................................... 68
  Object-Oriented Programming .................................................................................................. 70
  Efficiency of Algorithms ........................................................................................................... 74
2.2.3 Concluding remarks ........................................................................................................ 75
2.3 Affective dimensions of learning ...................................................................................... 75
  2.3.1 Defining affect .................................................................................................................. 76
  2.3.2 Motivation in education .................................................................................................. 78
  2.3.3 Self-efficacy and calibration ........................................................................................ 79
    2.3.3.1 Self-efficacy and achievement ................................................................................. 81
    2.3.3.2 Self-evaluation and Calibration ............................................................................... 83
    2.3.3.3 Self-efficacy in computer programming courses ..................................................... 84
  2.3.4 Task value, sense of belonging and identity ................................................................. 94
2.4 Concluding remarks ........................................................................................................ 99
3 Chapter 3: Research Methodology ...................................................................................... 102
  3.1 Philosophical Assumptions ............................................................................................... 102
  3.2 Methodological Approach ............................................................................................... 108
    3.2.1 Phase 1a: Identifying threshold concepts – The Delphi approach ..................... 108
      3.2.1.1 The Delphi method ................................................................................................. 109
      3.2.1.2 Characteristics of the Delphi study ...................................................................... 110
      3.2.1.3 Participants ............................................................................................................ 112
      3.2.1.4 Data analysis ........................................................................................................ 114
      Before the start of the Delphi rounds .............................................................................. 115
      Delphi – Round 1 ............................................................................................................. 115
      Delphi – Round 2 ............................................................................................................. 116
      Delphi – Round 3 ............................................................................................................. 116
      Determining consensus and stability ............................................................................. 117
      3.2.1.5 Researcher’s role and validity challenges ............................................................... 119
    3.2.2 Phase 2: Teachers’ perspectives and experiences with threshold concepts – An interpretative phenomenological analysis ................................................................ 122
      3.2.2.1 Interpretative phenomenological analysis ............................................................ 122
6.1.1 Case Description - Rea ................................................................. 168
6.1.2 Case Description – Andrea .......................................................... 171
6.1.3 Case Description - Olivia .............................................................. 178
6.1.4 Case Description - Mateo .............................................................. 182

6.2 Teachers’ experiences across cases .................................................. 185
6.2.1 IPA Analysis and Findings ............................................................. 186
6.2.2 Concept 1: Parameters ................................................................. 188
   6.2.2.1 Parameters are conceptually difficult ........................................ 188
   6.2.2.2 Skills needed to understand parameters .................................... 190
   6.2.2.3 Parameters and Conceptual changes and Coherence .................... 193
6.2.3 Concept 2: Parameter passing ....................................................... 195
   6.2.3.1 Parameter passing is difficult for students .................................. 195
   6.2.3.2 Overcoming parameter passing difficulties ............................... 199
   6.2.3.3 Parameter passing and Personal Transformation ....................... 201
   6.2.3.4 Parameter passing and Conceptual change and Coherence ............ 202
6.2.4 Concept 3: Return Values ............................................................. 204
   6.2.4.1 Return values are conceptually difficult .................................... 204
   6.2.4.2 Return values and changes in students’ understandings ................ 207
6.2.5 Group: Parameter, parameter passing and return values .................. 209
6.2.6 Concept 5: Procedural Decomposition .......................................... 215
   6.2.6.1 Conceptual and Practical difficulties ....................................... 215
   6.2.6.2 The value of decomposition and students’ flair for programming .... 218

6.3 Conclusion .................................................................................... 220

7 Chapter 7: Discussion of Phase 2 – IPA study ................................... 222
7.1 How do computing teachers experience the potential threshold concepts through their teaching and students’ engagement with these concepts? ................. 223
7.2 Is there evidence that supports the nomination of these concepts as threshold concepts, conceptions, or skills? .................................................. 225
   7.2.1 Parameters, parameter passing, and return values ....................... 225
   7.2.2 Procedural decomposition ......................................................... 229
7.3 Concluding remarks ..................................................................... 231

8 Chapter 8: Results of Phase 3a - Affective dimensions of learning and threshold concepts ................................................................. 233
8.1 Preparation of the data ............................................................................................................. 233
8.1.1 Interrater reliability .............................................................................................................. 236
8.1.2 Normality of the data ........................................................................................................... 238
8.1.3 Descriptive statistics ........................................................................................................... 242
8.2 Calibration in liminal and post-liminal space ......................................................................... 245
8.2.1 Descriptive statistics ........................................................................................................... 245
8.2.2 Correlations ........................................................................................................................ 246
8.2.3 General comparison between groups ..................................................................................... 249
8.3 Self-efficacy in liminal and post-liminal space ........................................................................ 251
8.3.1 Comparison between liminal and post-liminal group ............................................................. 251
8.3.2 Comparison within liminality groups and between reversed and non-reversed group 253
8.4 Task value in liminal and post-liminal space .......................................................................... 256
8.4.1 Comparison between liminal and post-liminal group ............................................................. 256
8.4.2 Comparison within liminality groups and between reversed and non-reversed group 258
8.5 Sense of belonging, motivation and identity in liminal and post-liminal space .................... 260
8.5.1 Comparison between liminal and post-liminal group ............................................................. 260
8.5.2 Comparison within liminality groups and between reversed and non-reversed group 262
8.6 Correlations with performance ............................................................................................... 268
8.7 Regression Model .................................................................................................................... 270
8.7.1 Assumptions ........................................................................................................................ 271
8.7.1.1 Part 1: predicting liminal or post-liminal space considering both students’ self-evaluations and the other affective variables ................................................................. 274
8.7.1.1.1 Model for the non-reversed group ............................................................................... 274
8.7.1.1.2 Model for the reversed group ...................................................................................... 280
8.7.1.2 Part 2: predicting liminal or post-liminal space considering only the affective factors apart from the self-evaluation ...................................................................................... 286
8.7.1.2.1 Model for the non-reversed group ............................................................................... 286
8.8 Conclusion .............................................................................................................................. 290
9 Chapter 9 - Results: Phase 3b – Students’ misconceptions in functions .............................. 292
9.1 Common misconceptions when students are in liminal spaces ......................................... 292
9.1.1 Interrater reliability .............................................................................................................. 292
9.1.2 Investigating students’ errors per task

9.1.2.1 Task 1 - Basic Understanding of function concepts: concept matching exercise

9.1.2.2 Task 2 - Knowledge of syntax and semantics in functions: find the errors exercise

9.1.2.3 Task 3 - Code comprehension and tracing with functions

9.1.2.4 Task 4 - Parsons puzzle with functions

9.1.2.5 Task 5 - Advance understanding - Fill in the blanks

9.1.2.6 Task 6 - Advanced understanding: Open-ended coding question

9.2 Summary

10 Chapter 10 - Discussion: Phase 3 – Affective dimensions of learning and misconceptions in threshold concepts

10.1 Calibration and liminality

10.2 Self-efficacy in liminal and post-liminal space

10.3 Task value and liminal space

10.4 Sense of belonging, motivation, and identity in liminal space

10.5 Regression model

10.6 Misconception in functions

10.7 Concluding remarks

11 Chapter 11 – Conclusion

11.1 The thesis and main findings

11.2 Research contribution

11.2.1 Methodological insights

11.2.2 Threshold Concepts Framework

11.2.3 Computer programming education

11.3 Limitations

11.4 Critical reflection

11.5 Future research directions

12 Bibliography

13 Appendix

13.1 Consent form for teachers

13.2 Information sheet: Phase 1

13.3 Questionnaire phase 1: Delphi study
13.4 Questionnaire phase 2: IPA study .................................................................417
13.5 Information sheet phase 3.................................................................418
13.6 Questionnaire and programming test phase 3 ..................................420
  13.6.1 Tasks and Bloom and Solo level.................................................428
13.7 Ethics Approval Phase 1 and 2.........................................................429
13.8 Ethics approval – phase 3.................................................................431
13.9 Published Papers based on this thesis...................... Error! Bookmark not defined.
1  Chapter 1: Introduction

Many studies in computer programming have been conducted at the undergraduate level, focusing on the factors that impact students’ performance and attitudes in this discipline. In computer science departments, this research has resulted in the general conclusion that undergraduate programs are complicated and challenging for students and that one of the most challenging courses in computer science is computer programming, which is often associated with high rates of failure and drop out in the departments of higher education (Bennedsen & Caspersen, 2007; Bergin & Reily, 2005).

Computer programming has become part of the curriculum in secondary education across many countries. However, writing computer programs is a complex cognitive ability (Kagan, 2006) and thus identifying and tackling learning obstacles is critical for students’ progress in the discipline. The literature has emphasised that students often struggle with different aspects of computer programming (Milne & Rowe, 2002; Beaubouef & Mason, 2005; Bennedsen & Caspersen, 2007; Bergin & Reily, 2005). Among the factors that impact students’ performance are the conceptual difficulty of computer programming, the feedback that students get from programming tasks, the patterns of study, and the lack of problem-solving, reasoning, analytical, planning and algorithmic skills (Butler & Morgan, 2007; Sarpong, Arthur & Amoako, 2013).

Focusing on the conceptual difficulties that students face in programming, the studies conducted since the mid-1980s have influenced and set the basis for further research in this area. These studies highlight fragile knowledge (Perkins & Martin, 1986), misconceptions and misunderstandings (e.g. Du Boulay, 1986; Perkins & Simmons, 1988) as sources of
students’ conceptual difficulties. More recently, the work of Eckerdal and her colleagues (Eckerdal et al., 2005; 2006; 2007) extended this list by focusing on the role of threshold concepts on students’ conceptual understandings and practice of programming (e.g. Thomas et al., 2017). Undoubtedly, understanding students’ imprecise conceptualisations is extremely important and a prerequisite for identifying and designing effective teaching methods both in undergraduate and school settings.

In this thesis, the focus is on the conceptual difficulties that students, at key stage 4 and 5\textsuperscript{1}, encounter in computer programming and specifically, on the threshold concepts and misconceptions that students experience in functions. Fleury (1991) and Madison and Gifford (1997) were among the first who highlighted the difficulties that students encounter with concepts such as arguments, parameters and parameter passing, all being part of functions in programming. Going a step further, Miller et al. (2015) suggest parameter passing as a potential threshold while procedural abstraction was included in a list of 33 concepts in programming that are potential thresholds (Boustedt et al., 2007).

Given the conceptual complexity of this area in programming, the study sets functions in the centre of the investigation. The thesis is grounded on the threshold concept framework and misconceptions and attempts to explore and understand students’ difficulties in programming under this perspective and framework as well as the contribution of affective constructs on students’ progress in the discipline. In the following subsections of this chapter, I first present the background of the research, the role of programming in the school curriculum in England, and the reasons the curriculum was transformed in primary and

\textsuperscript{1} Key stage 4 and 5: students are aged between 14-18
secondary education. I then describe the research problem, aims, questions, and how this research was designed to address these questions effectively.

1.1 Background to my research - Curriculum transformation

In today’s globalised economy and society, where access to information is fundamental and essential for knowledgeable individuals, schools and more generally, educational structures cannot remain detached or disconnected from the society’s fast-tracked changes. To this end, Information and Communication Technology literacy has been one of the most important goals for school education systems in many European countries. Nevertheless, European research has indicated that there is a growing gap for ICT experts in Europe; specifically, the estimation is that this gap could approximate 900,000 by 2020 (European Commission, 2014 – Digital inclusion and skills in the EU, 2014).

Thus, in the last decades, concerns have been raised about the suitability of the ICT curriculum regarding young people’s skills. Most of the European countries have come to realise that the education system, regarding Computing, was not adequate and did not respond to the needs of a digital society. The education system, concentrating mostly on the use of the technological application, was failing and should instead turn its focus to the founding principles of Computer Science and how these applications actually work (Jones et al., 2013). For this reason, many countries have transformed their curriculum by introducing Computer Science as a school subject in place of ICT. One of these countries is England.

In the early 2000s, computing education was declining in England. The school computing curriculum was focusing on Information and Communication Technology, which included
mainly the teaching and learning of office software (Grout & Houlden, 2014). According to Brown et al. (2014), ICT’s reputation among students was weakening as the students felt that it was a boring subject and was regarded as a low-value discipline. Even worse, students usually regarded ICT as the same subject as Computer Science and ended up abhorring what they erroneously thought Computer Science represents (Brown et al., 2014).

In the last years, there has been a turn-around in the focus of Computer Science education in primary and secondary education; the curriculum has shifted, and the teaching of ICT applications was replaced by rigorous academic computing courses (Hubwieser et al., 2014). This change was stimulated by the pressure of a variety of groups both from the industry and interest groups which aimed at persuading the government to realise the value that a computer science curriculum would have for students. Specifically, in 2008, Computing at School (CAS) was formed to reverse the disappearance of Computer Science’s presence in schools. CAS role was and still is so vital that it is regarded as the official subject association for Computer Science in the UK (Brown et al., 2014). By 2012, many other organisations joined this initiative. Particularly, the Royal Society (UK’s Academy of Sciences) published a report (Shutdown or Restart: The way forward for computing in UK schools, [Royal Society, 2012]) in which they included recommendations regarding the reintroduction of Computer Science. More recently, the report “After the reboot: computing education in UK schools” captures the changes that occurred since 2012 and discusses their impact and challenges of the digital world (Royal Society, 2017).
1.2 My motivation in this research

My interest in conducting this study stemmed from my experience as a computing teacher. I taught computer science and programming for six years in different types of schools in Greece, such as secondary schools, and vocational schools. During these years, I had experienced students who excelled in computer programming and considered it an “easy” subject while others struggled to understand it and often gave up on the subject. I admit that this was a challenging situation that often made me question my teaching practice and approach. Schön (1983: 68) argues that when a practitioner experiences a puzzling situation, like the one above, he “reflects on the phenomenon before him, and on the prior understandings which have been implicit in his behaviour. He carries out an experiment which serves to generate both a new understanding of the phenomenon and a change in the situation”. Schön called that reflection in action. Thus, reflection in action made me wonder what has triggered this event and how I could modify my teaching and methods to address this situation.

As a reflective teacher, I often got involved in reflective practices. Reflecting on my teaching practice was helpful for me to recognise my students’ problems and therefore, change my teaching approach in a way that reflects my students’ needs. Dewey (1933) argues that reflective action “involves a willingness to engage in constant self-appraisal and development. Among other things, it implies flexibility, rigorous analysis and social awareness” (Dewey, 1933 cited in Pollard, 2005: 13). The reflective operation, apart from the belief or doubt in something, also embraces further methods that can drive teachers into an enquiry of collecting evidence that will sustain their belief or disbelief (Dewey, 1910). For me, thus, it was necessary to search for research that would address my considerations.
and would explain the reasons why students encounter difficulties in programming as well as the sources of their misunderstandings. Most importantly, I was interested in research that would suggest approaches that I, as a teacher, could adapt and that could help my students cope with their difficulties. It was during this reflective inquiry that I came across two of the most studied conceptual problems that the students face in this subject: Misconceptions and Threshold Concepts.

While there is a considerable amount of research in both these areas (Misconceptions and Threshold concepts), I find that both areas need further investigation in two particular directions: firstly, on exploring how students experience these problematic areas and secondly, on the examination of teaching strategies that could help address the students’ conceptual problems. Particularly in the area of threshold concepts, most of the studies in programming concentrate on identifying these concepts. However, the existing studies do not explore affective constructs and how these differ among students nor make suggestions on how teachers can support their students to surpass the difficulties. This gap was my greatest motivation for researching this field further. My own experience as a teacher and the problems I encountered were the driven forces to extend the current framework of threshold concepts in computer programming in two ways: firstly, to identify potential threshold concepts and misconceptions in computer programming for students in secondary education, and secondly, to deeply understand why these concepts are experienced as thresholds, and how affective dimensions of learning and motivational constructs are impacted when students are in a liminal space.
1.3 Research Problem and Context

1.3.1 Research problem

Many research studies advocate the difficulty of computer programming. Most of these studies focus on the undergraduate level and less focus on the school curriculum. However, the need to address at the school level questions and challenges that researchers in undergraduate departments have identified, it is greater than ever now that programming is a fundamental part of the school computing curriculum. Thus, challenges like the conceptual difficulties students encounter in first-year computer programming, the misconceptions that they form, and threshold concepts they come across, are important to be investigated as early as the students engage with computer programming.

The study that is presented in this thesis aims to address the above argument. To achieve that, the study is focusing on students’ conceptual difficulties in functions in computer programming but at the school level and specifically at key stage 4 and 5. In this research area, studies in undergraduate level revealed that students form misconceptions from the beginning of programming instructions (the literature chapter provides a comprehensive review of these studies). Additionally, research suggests that in every discipline, there are some concepts posing extra difficulties for learners. Meyer and Land (2003) characterise these concepts as thresholds and they argue that their understanding is critical for mastering a discipline.

Many studies have been conducted with the aim of identifying threshold concepts in different disciplines. In computer programming, research studies focus on the undergraduate level and identify threshold concepts in the broader area of computer programming. For example,
Boustedt et al. (2007) recognise that the thresholds they have identified, and especially Object-Oriented Programming, are broad areas in which other thresholds exist. While research in this area revealed potential threshold concepts in computer programming, some of these concepts are too broad to call for a pedagogical intervention and also refer to undergraduate students. Thus, a narrower thematic area to identify these concepts would be more efficient, especially when the researcher is planning to explore and suggest ways or strategies for overcoming the conceptual and affective barriers that these concepts impose on students. Moreover, conducting research in secondary education settings may also reveal threshold concepts that have not been identified at the undergraduate level.

Thus, the research problem of this study focuses on identifying potential threshold concepts as well as misconceptions in computer programming at key stage 4 and 5, specifically in the area of functions, and on investigating how students experience these threshold concepts by putting in the centre of the investigation the affective domain. The reasons why this investigation has narrowed down the area of exploration to that of functions in computer programming are the following: Firstly, this is an area in programming that research has characterised as challenging for students, an area in which students experience misconceptions, and an area that may be a threshold for students (e.g. Fleury, 1991; Madison and Gifford, 1997; Boustedt et al., 2007). Secondly, drawing on my experience as a teacher, I believe that functions are conceptually and procedurally difficult for students, and therefore, a fertile area for thresholds and misconceptions to arise. I should note at this point that in this thesis, the concept of function refers to a block, a set of statements that is used to perform an action, a computation based on a given set of parameters, generating an observable output. Therefore, the thesis considers functions as abstractions over expressions.
In this research study, two groups of participants were considered: students and expert teachers. The reason for including this particular sample is that together these different levels of expertise can provide more insights to the threshold concept framework: students struggle to overcome conceptual difficulties in computer programming and often get stuck and frustrated with this discipline. On the journey from novice to expert, experienced teachers can provide valuable assistance; they can help their students surpass their difficulties by directing their teaching more on the curriculums’ difficult parts and highlighting ways to overcome the conceptual obstacles of programming.

1.3.2 Research Statement, Questions and Methodology

Functions and decomposition in computer programming are particularly challenging for students. Independently of the source of these challenges (e.g., conceptually and procedurally difficult knowledge), students’ engagement with this part of the programming curriculum opens a new knowledge space that it may challenge students’ belief system. The thesis argues that mastering functions and decomposition in computer programming engage learners with different forms of troublesome knowledge and ontological and epistemological shifts during which a number of affective constructs are impacted, questioned and negotiated, highlighting new avenues both for the threshold concept research and computer programming education.

The thesis explores this argument under the threshold concept framework. The overall aim is to identify potential threshold concepts and misconceptions in functions in computer programming and to explore some of the affective factors linked with liminal space. The thesis endeavours to contribute to the understanding of challenges that students face when
mastering functions and decomposition, their potential impact on students’ emotional capital, and the ontological, epistemological and affective shifts that are part of this journey.

To this end, the current study is separated into three distinct phases and eight research questions.

1.3.2.1 Phase 1: Identification of potential threshold concepts: a Delphi study

This first phase\(^2\) of the study focuses on identifying potential threshold concepts in functions. Specifically, this phase of the research aims to address the following research questions:

1. What do computing teachers report about students’ difficulties in computer programming at key stage 4 and 5, and what are their perspectives on the usefulness of the threshold concept framework in secondary computer programming?

2. What are the concepts that computing teachers agree as being threshold concepts in the area of functions?

To answer the first research question, a multiple-choice questionnaire was used while for the second question, the Delphi Method was employed. This method is known as a consensus method and is employed when the researcher seeks to achieve consensus among the participants. The panel that took part in this study was made up of 10 computing teachers; 6 of them were also practitioners with experience practising programming outside school settings for more than seven years. Three online questionnaires were sent individually to all

---

\(^2\) The results of the first phase of the research were published in the following paper which can be found in the Appendix: Kallia, M. and Sentance, S., (2017). Computing Teachers’ Perspectives on Threshold Concepts: Functions and Procedural Abstraction. In Proceedings of the 12th Workshop on Primary and Secondary Computing Education (pp. 15-24). ACM
the participants, and an iteration process of design, distribution, data collection and analysis was used. The data were qualitatively and quantitatively analysed with the SPSS software.

Based on this study’s findings, it was evident that further research was needed in order to expose the transformative and integrative characteristics of the concepts proposed as thresholds. The second phase, which is described in the following section, explores in-depth how teachers experience in their practice the transformative and integrative characteristic of these concepts.

1.3.2.2 Phase 2: Teachers’ experiences with potential threshold concepts: An interpretative phenomenological analysis

The previous part of the study resulted in a list of 11 concepts that computing teachers agreed on being potential threshold concepts. However, to enrich these findings, it was necessary to conduct further interviews with the teachers. The second part of the study\(^3\) aimed to explore how teachers experience these concepts in their practice and most importantly, the teachers’ experience seeing their students being transformed by grasping these concepts. Specifically, this phase aims to answer the following research questions:

3. How do computing teachers experience the potential threshold concepts through their teaching and students’ engagement with these?

4. Is there evidence that supports the nomination of these concepts as threshold concepts, skills or conceptions?

\(^3\) The results of this phase were published in the following paper: Kallia, M. and Sentance, S., 2020. Threshold concepts, conceptions and skills: Teachers' experiences with students' engagement in functions. Journal of Computer Assisted Learning.
To answer these questions, this phase employed the Interpretative Phenomenological Analysis (IPA) method. IPA draws from three methodological areas that of phenomenology, hermeneutics and idiography that are combined and form the IPA study. All these three methodological areas combined offered new insights into the threshold concept framework and in how teachers and students experience these concepts in their practice and learning, respectively. Semi-structured interviews were used as the method of data collection, and the sample consisted of 4 teachers. The interviews lasted approximately 40 to 90 minutes, and the transcripts were qualitatively analysed.

1.3.2.3 Phase 3: Affective dimensions of learning, misconceptions and threshold concepts

The third phase of the research turns the focus on students. The study here aimed to explore the affective factors linked to liminality, and the misconceptions students hold in functions. The liminal space, according to Meyer and Land (2003), is a situation of partial understanding where the learner fluctuates between the known and the unknown and where the learner experience emotions of troublesome, uncertainty and anxiety in his/her attempt to make meaning of a new concept or experience. To address and learn more about how students experience this phase, and therefore, to help teachers support students when passing through this phase, we need first to identify the relationship between the affective factors and students’ passage through these insecure spaces. Specifically, the study focuses on

---

4 Some of the results of this phase were published in the following papers and can be found in the Appendix: Kallia, M. and Sentance, S., 2019, February. Learning to use Functions: The Relationship Between Misconceptions and Self-Efficacy. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (pp. 752-758). ACM
students’ self-efficacy, calibration and self-evaluation, task value, sense of belonging, computer science identity and motivation in computer programming. The questionnaires that were used to address the above factors are the MSLQ developed by Pintrich et al. (1991) and the BASICS Study ECS Student Implementation and Contextual Factor Questionnaire Measures (Outlier Research & Evaluation, 2017). The computer programming tasks were designed according to Bloom’s and SOLO taxonomy.

Specifically, this phase attempts to address the following research questions:

5. Does being in liminal or post-liminal space affect students’ calibration, self-efficacy and task value in computer programming, sense of belonging, motivation and identity in computer science class?

6. Does students’ encounter with troublesome knowledge impact their calibration, self-efficacy and task value in computer programming, sense of belonging, motivation and identity in computer science class?

7. Can a model be used to predict students that are in liminal or post-liminal space?

To identify students’ misconceptions in functions, the students’ responses in the programming tasks were qualitatively analysed. The specific research question addressed by this phase is the following:

8. What are the misconceptions of 16-year-old students in the area of functions?

1.4 Overview of the thesis

The structure of my thesis reflects the three phases of the study and, thus, most of the chapters follow a three sub-section structure, one for each of the phases. In detail, chapter 2
is a review of the literature on students’ conceptual difficulties in computer programming such as misconceptions and threshold concepts as well as an exploration of the role of the affective dimensions of learning in students’ performance in general and in computer programming. The next chapter (chapter 3) presents the methodological framework of the data collection and data analysis for each of the three phases of the research. Thus, the chapter first highlights the philosophical assumptions that guide the current research with an emphasis on my ontological and epistemological beliefs. I then provide a thorough description and rationale of employing the diverse methodological approaches, data collection and data analysis techniques for each phase of the study. The chapter concludes with my role as a researcher, the validity issues in each phase of the study and the ethical considerations.

Chapters 4-10 present the analysis of the data collected in each phase and the discussion of the results. Particularly, chapter 4 outlines the Delphi approach of the first phase of the study and the corresponding results which refer to the concepts that reached a consensus of being potential threshold concepts in functions. Chapter 5 continues with the first phase again and discusses the results presented in the previous chapter. Chapter 6 examines the teachers’ experiences teaching the threshold concepts identified in the previous phase of the research and explores how each of the participants views the transformative and integrative characteristic of these potential threshold concepts. Chapter 7 provides a discussion of the results presented in chapter 6 and provides answers to the corresponding questions of the second phase of the study. Following this chapter are chapter 8 and 9, which focus on presenting the results regarding the students’ affective dimension of learning when they are in liminal and post-liminal space and students’ misconceptions correspondingly. Chapter 10
brings together these findings and discusses the results by combining the theory discussed in the literature chapter, together with the findings presented in chapters 8 and 9.

The thesis concludes with chapter 11 by drawing together the key findings and exploring these in light of the research questions of this research. The chapter also discusses the research contributions and the research impact on the pedagogy of computer programming and provides a reflective critique of the topic and focus of this research and possible future directions.
2 Chapter 2: Literature review

This chapter aims to provide the theoretical background of this thesis. The chapter starts by defining and discussing threshold concepts and relevant research conducted in this area as well as misconceptions in computer programming. The chapter concludes by presenting the role of affect in learning and particularly focuses on self-efficacy, task value, sense of belonging, identity and motivation.

2.1 Threshold concepts

This section aims to elucidate the theory of threshold concepts as it was presented by Meyers and Land (2003). Firstly, the section defines the terms concept, threshold concepts and liminality, and then moves forward to present and describe the characteristics of threshold concepts which make them different from other concepts in a discipline. Furthermore, it describes the methodology employed so far to identify these concepts in several disciplines. A critique is also provided of the methods other researchers employed and of ambiguous issues that arise from the theory as presented thus far. The section finally presents the most significant research studies conducted to date in computer programming and other disciplines regarding threshold concepts.

2.1.1 Threshold Concepts and Liminality

The term concept has been used in a variety of ways in the literature. In this thesis, I follow the psychological view that sees concepts as mental representations. Edward Smith
characterises a concept as “a mental representation of a class or individual and deals with what is being represented and how that information is typically used during the categorization” (Smith, 1989:502 cited by Goldstone & Kerten, 2003). A concept, therefore, denotes a mental idea or notion, and categories are entities that have been congregated together (Goldstone & Kerten, 2003).

Within the theory of threshold concepts, the curtain opens with the work of Eric Meyer (director of the Centre for Learning, Teaching and Research in Higher Education, University of Durham) and Ray Land (director of the Centre for Teaching, Learning, and Assessment, University of Edinburgh). Their work commences in 2003 with the Enhancing Teaching and Learning Environments project which aimed at detecting factors that can decidedly contribute to high-quality Higher Education learning environments within five disciplinary contexts. In particular, they introduced the notion of threshold concepts, concepts that form an interrelated web and are significant for mastering a discipline (Davies and Mangan, 2006).

According to Meyer and Land (2003:1), a threshold concept is defined as

“akin to a portal, opening up a new and previously inaccessible way of thinking about something. It represents a transformed way of understanding, or interpreting, or viewing something without which the learner cannot progress.”

The definition of threshold concepts emphasises the new conceptual landscape which the learner crosses where initial notions and ideas, formerly not understood, come into sight. Once understood, threshold concepts lead to a qualitatively different way of seeing a discipline and the learning experience (Kiley & Wisker, 2009). That is the basic characteristic that distinguishes a threshold concept from a core concept. Explicitly, Meyer
and Land (2003) argue that the main difference between a core and a threshold concept is that the former is a conceptual building block, essential to be understood in order for progress in the course to be achieved, whereas threshold concepts’ understandings lead to a qualitatively different way of seeing the subject. This indicates that a threshold concept diverges from other categories of concepts due to its transformative characteristic - an important change in the insights of a discipline - which core concepts lack. The time of this transformation, according to Meyer and Land (2003), maybe abrupt or prolonged over an extended period during which the learner will experience emotions of troublesomeness. However, the successful transition will enable learners to look at their subject in new ways and to act like their disciplines’ specialists.

Underneath the foundations of threshold concepts is the notion of liminality. The word liminality stems from the Latin word “limen” and liminality means ‘a suspended state’ in which understanding is based on mimicry (Meyer & Land, 2003:13). According to Hawkins and Edwards (2015), the concept of liminality derives from the discipline of Anthropology and mainly from studies on a variety of tribal rituals in which existing relationships and practices are paused for a while. It is for this reason, Delanty (2010:31) comments, that liminal moments have been defined as ‘moments in and out of time’ (Delanty, 2010:31, cited to Hawkins and Edwards, 2015). In other words, when a learner experiences difficulty in understanding a concept, she might find herself in a state where understanding is based on imitation or absence of authenticity (Meyer & Land, 2003). It is an insecure space where the learner fluctuates between old and new understandings. As Cousin (2010) aptly explains, liminality is a space just like the one that a child passes to move to adulthood.
Land et al. (2014) highlight that the liminal space entails both a conceptual and ontological change. From their viewpoint, this space can be regarded as a progressive function which is triggered when the learner comes across something new (Land et al., 2014). From that point, the learner should get involved in the process of identifying her deficiencies in her existing understanding of the thematic area under investigation and ultimately derogating from the older predominant view. Subsequently, this will require unlocking of the learner’s earlier style of subjectivity (Land et al., 2014). A very well-expressed point of view is that of Baillie et al. (2012) who consider liminal space as ‘heterotopia’, a place where alternatives are considered, and common sense is questioned.

The characteristics of this passage are well summarised in the work of Eckerdal et al. (2007): the liminal space is a space in which someone exhibits a transformation, an acquisition of “new knowledge and subsequently a new status and identity within the community of practice” (Eckerdal et al., 2007:124). Of course, this is not an easy passage. It requires time and includes feelings of nervousness, eagerness, and difficultness. Meyer and Land (2003) recommend threshold concepts as the key to enter and pass through liminality.

2.1.2 Characteristics of Threshold Concepts

Not all concepts in a discipline can be categorised as threshold concepts. To be able to categorise a concept as a threshold, Meyer and Land (2003) presented seven key characteristics of a threshold concept that are discussed in the following paragraphs.

The first of the characteristics of a threshold concept is its transformative nature. This characteristic denotes that once these concepts are understood, they evoke an important change in students’ behaviour and in how students perceive their disciplines. Meyer and
Land (2003) also believe that once these concepts are understood, a potential effect is to occur a transformation of learners’ personal identity which involves a change in values and beliefs, emotions or attitudes. As explained by Cousin (2006), understanding a threshold concept is a transformative process as it implicates both an ontological and a conceptual change. Without this shift in learner’s perception, progress cannot be achieved. In fact, this characteristic of the threshold concepts is so dominant that it can be regarded as a super-category in which the other characteristics of threshold concepts are gathered (Walker & Guy, 2013).

Threshold concepts are probably irreversible. This signifies that the shift on learners’ view of the subject and the modification of their perspectives occurred by understanding a threshold concept, are implausible to be forgotten. Additionally, threshold concepts are integrative. Actually, they not only lead to a transformation of one’s perception and understanding in a discipline but also, they can influence and change how other disciplines are viewed by integrating with the already existing knowledge (Sandri, 2013). In other words, they uncover the “formerly unseen interrelatedness of something” (Meyer & Land, 2003:5).

Moreover, threshold concepts are bounded. This characteristic indicates that they have borders that, when traversed, they can lead to other conceptual developments (Kiley & Wisker, 2009). Thus, these boundaries function as distinction points between subject areas, to define, as Meyer and Land (2003) aptly comment, academic regions.

Attributed to their transformative nature, threshold concepts can be puzzling and difficult to understand and consequently can be troublesome for students who engaged with these concepts. David Perkins (1999) describes this as something that seems counter-intuitive,
alien or incoherent. In fact, Perkins (1999) extensively refers to troublesome knowledge as a theory in order to elucidate the ways in which knowledge can be challenging for students. In detail, Perkins (2007) identifies five types of troublesome knowledge: Conceptually difficult knowledge that refers to knowledge that is hard and problematic to learn; Ritual knowledge that is like a routine but is futile; Inert knowledge that is knowledge passively invoked; Alien knowledge that is counter-intuitive to learners own perspectives; and Tacit knowledge which refers to essential and central rules of a discipline but veiled or instinctive as well. Perkins (1999) suggests constructivism as a theory to deal with troublesome knowledge and urges learners to undertake an active role in their learning experience. The troublesome characteristic of learning which leads to development has also been mentioned by Merizow (2000) who suggested that for transformation to occur, a learner must experience feelings of disorientation (Merizow, 2000 cited in Timmermans, 2010).

Finally, threshold concepts have two more characteristics: discursive and reconstitutive (Barradell, 2013; Flanagan, 2015; Land et al., 2014). The discursive characteristic refers to the change the learner undergoes crossing the threshold concept that leads her to express herself in more discipline-like ways (Harrison et al., 2014). In other words, crossing a threshold concept will improve the way the learner uses the discipline’s language. The reconstitutive characteristic refers to the change of the learner’s identity and subjectivity once the threshold concept is understood. It involves a reconfiguration of the learners previously formed schema (constructivism centred), which will cause an ontological and an epistemic shift (Land et al., 2014).

To conclude, Meyer and Land (2003) emphasised that it is possible that the discipline’s language, the learner’s diverse backgrounds and understandings may control the way
someone experiences concepts. That is why a concept may be easy for someone to comprehend, while at the same time it is extremely difficult for others who struggle to cross the liminal space.

2.1.3 Threshold conceptions and Procedural thresholds

Perkins (2006) proposed that the difficulty of some concepts may not stem from the concepts themselves, but from the way some concepts interact with each other to create an underlying game which causes a deep transformation on students’ understandings. For example, Land et al. (2005) explain that in computer programming the concepts of “class, objects, tables, arrays and recursion may not have the troublesome or transformative characteristic but what is troublesome and transformative for students is the way that these concepts fit together and interact in a process of ever-increasing complexity” (Land et al. 2005:56). These concepts were characterised as threshold conceptions for they "bind together aspects of a subject that may seem quite disparate to a novice" but "are fundamental to ways of thinking and practising in that discipline" (Land et al. 2005).

The threshold concept framework assumes that a phenomenon may be realised in many different ways and each one of them is a conception. Burch et al. (2015) provide a figure of conceptual development (figure 1) in which they demonstrate that knowledge in topics lead to the formation of concepts and when some of these concepts are integrated in a way that corresponds to a discipline’s ways of thinking then conceptions are formed. Based on this model, Burch et al. (2015:483) argue that students come across threshold concepts or conceptions when:
a. “They do not understand all the concepts to make the conception” (Yip and Raelin, 2011 cited in Burch et al., 2015)

b. “They are missing one or more concepts in their conception” (Hibbert and Cunliffe, 2013 cited in Burch et al., 2015)

c. “They do not properly integrate the concepts” (Hawkins and Edwards, 2013 cited in Burch et al., 2015)

d. “They do not view the concepts through the lens of the discipline” (Wright and Gilmore, 2012 cited in Burch et al., 2015)

![Figure 1: Conception Development (Burch et al., 2015: 483)](image)

Procedural or modelling thresholds were mentioned by Davies and Mangan (2007). In their work, they specifically identified three types of conceptual change: basic, discipline and
procedural or modelling (Table 1). They define threshold concepts in the discipline and procedural conceptual change as "understanding of other subject discipline ideas integrated and transformed through acquisition of theoretical perspective" and "ability to construct discipline-specific narratives and arguments transformed through acquisition of ways of practicing", correspondingly (Davies and Mangan, 2007:715). These procedural thresholds may be linked with threshold skills as proposed by Thomas et al. (2017) who reflect on skills as a form of procedural knowledge "difficult or impossible to write down and difficult to teach best taught by demonstration and best learned by practice" (Norman 1990 cited in Thomas et al., 2017:335) and they suggest five key features of a threshold skill: transformative, integrative, troublesome, semi-irreversible, and associated with practice.

*Table 1.1 Definition and exemplification of three types of conceptual change (Davies and Mangan, 2007:715)*

<table>
<thead>
<tr>
<th>Type of conceptual change</th>
<th>Type of transformation and integration</th>
<th>Examples in economics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Basic</td>
<td>Understanding of everyday experience transformed through integration of personal experience with ideas from discipline.</td>
<td>Distinctions between price/cost; income/wealth (stocks/flows); nominal/real values; investment/saving. Newly met concepts such as real money balances, natural rate of unemployment.</td>
</tr>
<tr>
<td>2. Discipline threshold concepts</td>
<td>Understanding of other subject discipline ideas integrated and transformed through acquisition of theoretical perspective</td>
<td>Interaction between markets, welfare economics, opportunity cost</td>
</tr>
<tr>
<td>3. Procedural</td>
<td>Ability to construct discipline specific narratives and arguments transformed through acquisition of ways of practicing.</td>
<td>Comparative statics (equilibrium, ceteris paribus), time (short-term, long-term, expectations), elasticity</td>
</tr>
</tbody>
</table>

From my perspective, I recognise skills as being included in procedural thresholds. According to Rittle-Johnson and Schneider (2015), procedural knowledge or skills are defined as a series of steps, actions to accomplish a task – is the “knowing how” – the knowledge of the steps/algorithms that are necessary to solve a problem. However, procedural thresholds include more than an ability to perform a task; they include a transformation stemming from the acquisition of ways of practising. The procedural concepts, according to Davies and Mangan (2006), are ‘enablers’ with which students can accomplish a more comprehensive understanding of the threshold concepts (discipline thresholds).

2.1.4 Threshold concepts and pedagogical implications

Meyer and Land (2005) suggested nine considerations for the design and evaluation of courses when threshold concepts are considered.

The first one views threshold concepts as “jewels in the curriculum,” which refers to the transformative perspective of the learning experience. This characteristic is based mostly on the fact that these concepts can help identify critical points in the curriculum to which teachers should pay extra attention in order to help their students gain new understandings of their subject. They can also help teachers identify parts of the curriculum where students may encounter conceptual difficulties.
Meyer and Land (2005) also highlight the “importance of engagement” as the second consideration. Several studies address the significance of engagement in education (e.g., Newmann, 1992). In fact, based on Bloom’s (1956) research, Fredricks et al. (2004) successfully recognise three dimensions of student engagement: behavioural engagement, emotional engagement, and cognitive engagement, all of which are pertinent to the theory of threshold concepts. In the threshold concepts’ framework, student engagement addresses the importance of a learning experience that offers various forms of engagement to students, so they can potentially understand and perfect ways of thinking and practising that correspond to their community of practice (Meyer & Land, 2005). In other words, it refers to the challenges that course designers/teachers should create through students’ engagement to provoke the desired transformation. This transformation should help students not only to understand how a computer scientist thinks but to think like a computer scientist.

The third suggestion is “listening for understanding,” which refers to teachers’ attitudes toward listening to what a student does not know, what a student does not understand, and where a student gets stuck or experiences difficulties rather than only to what a student already knows (Meyer & Land, 2005).

The fourth consideration is the “reconstitution of self,” which holds that grasping a threshold concept may lead to a shift of one’s self regarding the discipline. This transformation, as exciting as might be, includes a period of anxiety as the learner proceeds from one to another uncertain situation. Consequently, special attention should be given in the curriculum regarding the design of a supportive liminal environment which acknowledges the discomfort (Meyer & Land, 2005) students undergo.
The fifth consideration is “tolerating uncertainty,” which recognises students’ need to be supported when they are in a state of uncertainty and must leave behind previous ways of understanding to safely go through the liminal space (Meyer & Land, 2005). Students must build meta-cognitive strategies, which, according to Efklides (2005), will determine if students will engage in threshold concepts or not. Therefore, this consideration suggests that teachers should help their students build the confidence needed to get through this difficult phase and employ mechanisms to help them self-regulate their learning. The sixth point, “recursiveness and excursiveness,” explains how threshold concepts require a recursive approach with which learners can deal with a problem by adopting a more holistic approach (versus a linear one) and exploring multiple alternative ways for understanding the problem. It also calls for the curriculum to be planned in a way that includes variation both in conceptual engagement and how outcomes are regarded. The learning process should also be seen as a journey (excursive) with a proposed route and outcome, but that also recognises that deviations and unforeseen outcomes will occur.

The seventh point, “pre-liminal variation,” suggests that it is a good practice when designing curricular to consider the reasons why some students manage more easily to pass through the liminal space while others struggle and get stuck. This could be attributed to many factors, one of which is students’ emotional and affective capital. Meyer and Land did not offer much information about this consideration, although they link it with the next one. So, the eighth-point, “unintended consequences of generic good,” suggests that previous pedagogy, though considered good, may be found to be impractical when threshold concepts are considered. For instance, Meyer and Land argue that the intention to make concepts simpler in order to help students’ understanding may be a dysfunctional approach to
threshold concepts and may lead students to ritualised knowledge or naïve understandings (e.g. misconceptions). Meyer and Land (2005) report that students are often unaware that their understandings do not align with the authorised versions, and that a threshold conception may require students to understand the difference between their own understandings (misconceptions) and the authorised understanding of threshold concepts. Perkins (2006) refers to this as “the underlying game” which is the final consideration and refers to the significance of the role that threshold conceptions play when designing a curriculum. It refers to a deeper level of understanding that goes beyond the comprehension of a single concept and to which course designers must pay attention.

To conclude, the intention of this section has been to highlight threshold concepts as a significant and challenging aspect of a discipline’s curriculum. However, to assess threshold concepts’ value in the curriculum design, we need first to detect them, which is not a straightforward process, a topic which is addressed in the next section.

2.1.5 Methodologies of Identifying Threshold Concepts in disciplines and Critique

The threshold concept framework provides a fruitful area for developing teaching principles and guidelines to improve students’ understandings and achievement in a discipline. Indeed, the conceptual challenge that students face when confronted with demanding concepts in a subject makes education researchers eager to explore what students need to understand within a subject in order to progress. Therefore, a considerable amount of research has been conducted involving threshold concepts in a variety of subjects. The disciplines that have been researched mostly are Economics and Engineering, while others include the disciplines of Mathematics, Statistics, Health Sciences, Chemistry, and Geographical Sciences.
Nevertheless, a complete research methodology to identify threshold concepts has not yet been developed. Methods that have previously been employed focus on a qualitative approach (observations, surveys, interviews, course tests and assignments) and include students as well as teachers/lecturers.

An astute observation is expressed by Quinlan et al. (2013) who distinguish the way a researcher searches for threshold concepts depending on the researcher’s epistemological stances. To clarify, they argue that those who adopt a positivist approach to research, assume that there is only one truth that needs to be discovered and as such, they are looking for certain threshold concepts integral in a discipline. On the contrary, when constructivists are looking for threshold concepts, they take into consideration the way teachers deliver their courses, the background of their students, and where teaching takes place. Undoubtedly, this is far more compound than the approach used by positivists.

Davies (2006) suggested two methods for identifying threshold concepts. The first one includes the engagement of two distinct disciplines and specifically the views in which these disciplines examine the same situation. The second approach, which is mostly used in the current literature, suggests that to identify threshold concepts the researcher should concentrate on people inside and outside of the community, that is, the different ways in which students and experts experience the situation (Rountree & Rountree, 2009).

While early research in this field focused on students’ experiences through phenomenological research, many researchers are sceptical of including students to address such a complicated phenomenon. Among them, Male and Baillie (2011:52) argue that threshold concepts can be identified through two sources: first students and secondly from "people whose experiences give them awareness of students’ experiences" such as teachers.
They further support that teachers cannot only report on the troublesome aspect of threshold concepts but also the most essential aspects which are the transformative and integrative characteristic of these concepts. Upon this issue, Shinners-Kennedy and Fincher (2013) align with the researchers above regarding the use of teachers and suggest that threshold concepts can be identified by interviewers who possess both the pedagogical and the content knowledge. They further support that research in this field has reached a dead end. They criticise the methods used for identifying threshold concepts in programming, emphasising that asking students about difficulties confronted in the past is an unreliable method. Finally, Zwaneveld et al. (2016) argue that this pedagogical and content knowledge cannot easily be found in undergraduate teachers; instead, they propose secondary teachers to this end.

In any case, the majority of the research studies is focused on interviewing teachers first and then on exploring students’ view. Various research studies fall in this category and are presented in the following paragraphs.

2.1.5.1 Interviews with Students and Teacher/Lecturers

One of the first and extensive studies in the field of Economics and threshold concepts is the work of Davies and Mangan (2007). For their study, they collected a variety of data by interviewing students and lecturers in Economics. The authors collected their first results by collecting and analysing students’ and lecturers’ responses in problems that resembled the ones that real economists consider. The underlying reason for this approach was that the authors veritably expected that, if threshold concepts exist, they would explicitly be mirrored in lecturers’ responses. On the contrary, students’ answers would be referring to these concepts in a much vaguer way. Even if the discursive characteristic of threshold concepts had not been added when this research was conducted, it is obvious that the researchers
particularly emphasised the difference between the language used by more experienced users, in this case, lecturers, and the incompetent use of the discipline-language of novices, in this case of students. Indeed, their results reflect their initial hypothesis about the differences in lecturers’ and students’ responses with the latter being linear, taking into account only one aspect of the problem with no clues of modelling or a clear procedure to address the problem. The next round of analysis included data from four past examination papers from the universities that participated in the study. The analysis of these data suggested that the concepts of “partial equilibrium, marginal analysis, cumulative causation, interaction between markets and opportunity cost” (Davies and Mangan, 2007:718) had the highest frequency of occurrence and, thus, lecturers regard them as important concepts for students’ understanding of the discipline. However, by exploring students’ answers on these tests, the researchers concluded that students’ understandings of these concepts were very low with an obvious inability to use them correctly and in the way their lecturers expect them to do. Based on this finding, Davies and Mangan (2007) continued by investigating students’ conceptual variation of economic analysis through an exploration of their (open-ended questions) perceptions about this concept and by asking them about the troublesome concepts of the curriculum. From the analysis of their results, the researchers suggested that the concepts of demand and supply appear to be both troublesome and transformative for students and that integration is a dominant aspect of the development of students’ understanding in Economics. Finally, they recommended that it is better to regard threshold concepts as a web of concepts within a discipline rather than as stand-alone concepts.

In the same vein, Male and Baillie (2011), as part of the design of a new Engineering Major for BSc at the University of Western Australia, started exploring initial threshold concepts
in Engineering of the first and second-year undergraduate course. They were interested in identifying concepts that are both transformative and troublesome. During the interviews, the participants (academics and tutors) were asked to propose potential threshold concepts, to explain the reasons why they have selected these concepts, to clarify the transformative way of thinking that students have difficulty with and, finally, to suggest ways for helping students overcome these difficulties. Additionally, the investigators employed two focus groups with students who were asked to identify prospective threshold concepts. The first group consisted of chemical engineering students, whereas the second group consisted of five senior engineering students who had experience with tutoring. Their choice of employing both undergraduate students and senior engineering students was indeed appropriate because the latter set of participants have both recent experience of being students and experience as tutors and, thus, they could provide multiple views. The final step was discussions with academics in engineering, students and engineering teachers, in the form of workshops. After the analysis of their data, the researchers were able to develop an inventory of threshold concepts which contributed to the design of the new curriculum in Engineering.

Along similar lines, one of the earliest works in the discipline of Engineering and threshold concepts is the study of Baillie, Goodhew and Skryabina (2006). The researchers conducted surveys with two hundred and fifty students from four universities in the UK. The students were asked to evaluate (from the least interesting concept to the most interesting) concepts from Mathematics and Mechanics and explain the reasons why they feel that these concepts are or not interesting. The investigators argued that with this method, that is, asking students about positive and negative attitudes towards particular concepts in Mathematics, it would
be possible for them to uncover some ways of understanding or to pass through liminality. The concepts were selected from the faculty as concepts that pose difficulties to students. After the analysis of the interviews, the researchers discovered that a complex number is perceived as the least interesting concept in Mathematics due to its difficulty, boredom, and irrelevance to the degree. What is more interesting is that they also compared students’ negative responses with students’ positives ones, and they found that they were opposite (routine/novel, irrelevant/applicable, difficult/understandable). They attributed these differences to the fact that the first group has not passed through the liminal space yet, whereas the other group passed through the thresholds successfully. Finally, the researchers suggested that to help students through liminal space, they should engage in fruitful discussions with those students who are on the other side of understanding, by exposing variation in student learning.

Likewise, another two representative studies in this category are Wimshurst (2011) in the field of Criminal Justice Education and Hoadley et al. (2015) in Finance. Both studies adopted similar methodologies as the aforementioned studies - interviews with students and/or interviews with academics. The results of the former study prove the division on students’ understanding of their subject of Criminal Justice Education and the suitability of the framework of threshold concepts on describing students’ views of their discipline. The latter study aimed at updating the current curriculum and pedagogical practices for the students’ benefit by identifying threshold concepts in the curriculum. Indeed, the researchers proposed seventeen concepts that are potential thresholds and need to be taken into extra consideration in the design of the curriculum.
It is true that many research studies follow the same line as the studies outlined in the above paragraphs. However, there are certain considerations that have been reported with these approaches. Particularly, Davies (2006) indicates that interviewing teachers is problematic because they will either focus on fundamental concepts or concepts that are only included in the curriculum. Furthermore, when it comes to students, especially novices, it has been reported that they usually neglect or avoid mentioning concepts that have not yet been understood (Carstensen & Bernhard, 2008).

2.1.5.2 Tests, Quizzes and on Field Observations

Other researchers have employed quite different approaches to identify threshold concepts. For instance, Shanahan, Foster and Meyer (2006) attempted to link the theory with concrete measures by examining students’ replies in a multiple-choice questionnaire which measured their understanding of the threshold concept of opportunity cost. The researchers also wanted to investigate the correlation of different student factors with the understanding of this concept. Their study indeed offers an alternative approach for identifying students’ variation in understanding of a threshold concept. However, the researchers recognise that their results are not as sensitive as they would like them to be in order to assess students’ development in understanding the concept. Students were asked to respond to a Likert scale questionnaire (Reflections on Learning Inventory - RoLi), which was designed to support students in understanding and enhancing their meta-learning capacity. The questions referred to what the students think the economy is, their insights about the role of market forces in determining prices and what students think economists do. The students also completed another online questionnaire which evaluated their understanding of economics (Test of Economic Literacy - TEL). The students complete each of these tests twice, one in the first
two weeks and again in the final two weeks of the semester. The researchers based their results on students’ overall score on these tests and on students’ score on specific questions that addressed their understanding of opportunity cost. They also considered the students’ answers in the tutorial quiz, their final exam scores and the final percentage they received in the course. Hence, after the analysis of their data and the consideration of a variety of other factors, their results indicate that the understanding of the concept of opportunity cost is determined more by students’ background and preferences, their prior experience and knowledge, rather than from what happens in the classroom. The results also support that students’ final performance is faintly related to their understanding of this concept. This finding corroborates Davies and Mangan’s (2006) assertion that there is an interconnected web of concepts that students need to understand to progress in their disciplines. Nevertheless, the researchers acknowledged the limitations stemming by employing this kind of measurement. However, their work suggests that it is possible, though challenging, to develop reliable measures to examine students’ acquisition of threshold concepts.

Apart from employing tests, other researchers used quizzes with other techniques to identify thresholds. An example of this is the study carried out by Worsley et al. (2008), which focuses on identifying the threshold concepts in high-level Mathematics. The researchers used documents, interviews with tutors, and surveys and quizzes with students to identify the thresholds. Specifically, the researchers designed a quiz for testing students’ conceptual (open-ended questions) and procedural knowledge (True or False) understanding. The survey included more students and included questions about troublesome concepts the students encounter in a specific part of the curriculum. From the analysis of their data, the researchers were able to identify three potential threshold concepts, namely, ordinary
differential equation, the technique of substitution, and multiple integration. The tutors’ interviews were used to confirm these findings and the troublesome areas that the students had indicated. Finally, the investigators commented that during these interviews, the tutors revealed that these findings were really valuable for them, and they add to their experience as teachers.

With a completely different but novel approach, Cartensen and Bernhard (2008) took the initiative of exploring threshold concepts in Engineering by observation within an engineering lab. They employed video analysis techniques to identify troublesome concepts that also are integrated and bounded. The researchers aimed at identifying observed difficulty that students encounter and accounted that to threshold concepts’ troublesome characteristic. Of course, this is contestable. Students’ difficulties may be attributed to many other factors other than the existence of a threshold concept. Nevertheless, it can be argued that the approach wins on the fact that it is an immediate approach and also recognises the importance of addressing threshold concepts for the students.

2.1.5.3 Phenomenography and Reflective essays

Other approaches adopted by researchers are phenomenography and reflective essays. Akerlind’s et al. (2010) research is a good illustration of a study that employed phenomenography. Particularly, the researchers set up a research project to explore students’ misconceptions in the first year of a course on Legal Education, as part of a funded project of the Australian Learning and Teaching Council. The researchers employed phenomenographic and action research methods in the Law discipline in order to build up a model for the design of Law curriculum that attends to students’ misconception. The concept that the researchers focused on was legal reasoning, which was proposed by the projects’
law group as the first threshold concept that a student needs to grasp. Specifically, the researchers designed an action research project with a sequence of questions and a problem situation in order to uncover the qualitatively different ways students experience the concept of legal reasoning. The analysis of the data was conducted phenomenographically. The research’s results indicate that there are four levels of variation in students’ understandings of the key features of legal reasoning. Conclusively, they pointed out that the new curriculum design intends to present a method for helping students to turn their attention to the critical features of legal reasoning identified by their research.

Similarly, Eckerdal and Thune (2005) explored first-year undergraduate students’ understanding of the concepts of object and class in computer programming by employing a phenomenographic approach. Specifically, the researchers were interested in exploring the qualitatively different ways that students understand these objects. After the phenomenographic analysis, the researchers concluded on three different ways of understanding. By applying then variation theory, they were able to identify the critical aspects that a student needs to discern to reach a full understanding of these concepts. Even though the researchers did not explicitly refer to these concepts as thresholds, a later study of Eckerdal et al. (2006) categorise these concepts as potential threshold ones.

Finally, a very different but interesting approach is Fouberg’s (2013) research study. The focus of this research was the identification of threshold concepts in world regional Geography. By adopting a qualitative approach, the researcher asked students to complete a journal assignment which included different material covered in class and a reflective essay analysing their own learning. Some of the students were told to employ metacognition and to include a threshold concept essay for this task. The results of the analysed data indicate
that students focused more on the integrative and then the transformative and finally, the counter-intuitive characteristic of geographic concepts. Some of these concepts were commodity chains, core periphery, migration and remittances. A major contribution of this study though is that the author suggests that metacognition is an extremely valuable process both for the teacher to understand his/her students’ understanding and integration of the discipline’s concepts but for the students too in order to construct new schemata and transform the way they perceive their subject. This outcome validates Land’s et al. (2005) argument of the significance of metacognitive skills for the transformation and the uncertainty of liminal space.

2.1.6 Threshold Concepts in Computer Programming

In the discipline of computer programming, a variety of approaches has been used to categorise concepts with which students experience difficulties. In this section, a review of previous work on threshold concepts in the area of programming is described along with the methodologies employed and the limitations of these studies.

The most concrete work in the area of computer programming has been conducted by a group of researchers in the field of Computing Education: Anna Eckerdal, Robert McCartney, Jan Erik Mostrom, Mark Ratcliffe, Kate Sanders, Carol Zander, Jonas Boustedt, and Lynda Thomas. Their work commences in 2006 with a paper which discusses the notion of threshold concepts and how this theory can be applied in Computer Science. At the same time, in this paper, they reflect upon other related theoretical areas such as “constructivism, mental models, misconceptions, breadth-first introductions and fundamental ideas” in Computing Education and explain how they link to threshold concepts (Eckerdal et al.,
2006:104). They further suggest two possible threshold concepts in programming, Abstraction and Object-Orientation, which seems to hold the characteristics of thresholds. However, their study does not include empirical evidence to support their claims. Instead, the researchers are based on personal views of why these concepts are experienced as troublesome, transformative, and integrative.

The group’s next attempt, which is more complete, includes an empirical investigation conducted to identify threshold concepts in Computer Science (Boustedt et al., 2007). The authors collected data from Computer Science instructors who got involved in an informal interview and a survey. The interviews took place in June 2005, at the Conference on Innovation and Technology in Computer Science Education (ITiCSE). In detail, the researchers interviewed thirty-six instructors from nine countries who were asked to make recommendations of concepts that they thought include the five characteristics of thresholds. However, the researchers accepted that the interviews were unstructured and with an informal style. Nevertheless, from these interviews, the investigators collected thirty-three concepts. The concepts that had been suggested with the highest frequency were: “levels of abstraction, pointers, the distinction between classes, objects, instances, recursion, induction, procedural abstraction and polymorphism” (p.505).

At a later point, the researchers sought to collect data more systematically. In November 2005, they employed a questionnaire to interview other conference researchers at the Koli Calling 2005 Conference on Computer Science Education in Finland. The results from the analysis were quite similar to the ones collected at ITiCSE. However, the authors noted that the interviewees seemed to focus more on the difficulty that these concepts imposed on students rather than the other characteristics of threshold concepts (Boustedt et al., 2007).
Thus, their study does not provide convincing evidence that the concepts proposed by the interviewees possess the characteristics of threshold concepts other than the troublesome.

In the next phase, the researchers turned from the instructors to students. Having identified a list of concepts both from the literature and their previous studies, the group started exploring whether these concepts are experienced as threshold concepts from the students’ point of view. They chose to collect graduate students’ perspectives on these concepts and not novices. Indeed, their choice was appropriate since novice students have not usually mastered challenging concepts. The group employed semi-structured interviews, and sixteen students from seven institutions participated in the study. The interviews started by asking students to suggest a concept with which they experienced some stuck moments but then became easier. From the students’ suggestions, the researchers chose one concept to investigate further. Particularly, they concentrated on the transformative, integrative, and irreversible characteristics of threshold concepts. The analysis of the interviews encouraged the selection of the following programming concepts: object-oriented programming and pointers. The researchers demonstrated parts of the interviews that corroborate their claims about the transformative, integrative, and irreversible characteristics of these concepts.

The focus of the groups’ third research paper was the liminal space in Computer Science (Eckerdal et al., 2007). The research group was interested in exploring if the liminality, as described by Meyer and Land (2003), can be useful in understanding students’ conceptual transformations while they practice programming. Specifically, they were concerned with investigating characteristics of the liminal space such as the time required for the transition, the oscillations that a student experiences, their emotions during that time, and evidence of mimicry. The researchers used the data they had collected from their previous work and
explored once more the transcribed interviews, searching for signals related to liminal space. In the authors’ own words, they called their analysis as a “triangular conversation between the researchers, the data, and the liminal space” (Eckerdal et al., 2007:126). They also emphasised that the results of their study are an interpretation of their data and their own experiences as teachers.

Nonetheless, the results appear to be quite interesting and leave much room for further investigations on the thematic areas they covered. One such observation is that the researchers found contradictory results with well-known and established theories of learning. For example, their research indicates that the different stages of understanding take place in parallel. Indeed, they observed that students oscillate between the theoretical and the practical parts of the curriculum. Another interesting result of this study was that students’ difficulties lay in different aspects of a concept and that students use different ways through the liminal space. This result has implications for the design of the curriculum as it suggests that there is not one way to teach specific topics that would be accepted from all students. Instead, the curriculum should focus on the aspects of a concept that are significant for students to gain an understanding of it. Moreover, the authors observed that the students’ difficulties may start from understanding the theoretical side of a concept and may reach a point where students encounter difficulties in the concept’s application in practice. Finally, they observed that some students experience emotions of hate and fear but also feelings of euphoria when they safely leave the liminal space and that they also experience mimicry at some point during the learning process.

The group’s next research direction was to support the threshold properties of the concept of object-oriented programming. To accomplish this, they used a different approach (Sanders
et al., 2008). They were interested in exploring the students’ beliefs of the most important object-oriented concepts and how they express the relationship between these concepts. They assembled seventy-one novice programmers who had been given a concept map illustrating the concepts of kitchen and dinner and the following assignment: “put the concept map here that starts with the two concepts class and instance with labelled arrows and other concepts that creates a partial map of object-oriented programming, as you have learned it so far” (Sanders et al., 2008:333). The outcomes suggest that students face difficulties in understanding the meaning of class, object, and instance. Unfortunately, the results are questionable regarding their position in the framework of threshold concepts. Fincher and Kennedy (2013) challenged these results stating that although they indicated the existence of students’ misconception on concepts like class, object and instance, they do not lead to the conclusion of object-oriented programming being a threshold concept.

The next project of the research group focuses on the transformative characteristic of threshold concepts. Specifically, the study aimed to explore how abstraction is demonstrated in students’ experience (Moström et al., 2008). To address this question, the researchers collected transformation biographies from Computer Science students. They interviewed eighty-six students from five institutions (Sweden, United Kingdom, and the United States) who “were asked to identify and describe a computing concept that transformed the way they see and experience computing” (p.127). Of the eighty-six biographies, forty-seven referred to a concept that is related to the notion of abstraction. The analysis of these data (divided in seven general topics: “modularity, data abstraction, object-oriented concepts, code reuse, design patterns, complexity, and other concepts”, Moström et al., 2008: 127), led the group to suggest that abstraction, as discussed in their paper, is a threshold concept.
Overall, the work of this group is regarded as the most significant research in the area of computing programming and threshold concepts. Most recently, the investigators turn the research interest to threshold skills (Sanders et al., 2012). The researchers have identified that students often find difficulty on how to apply their knowledge to problems and not with the meaning of a concept. This suggestion has also been supported by other researchers such as Baillie, Bowden and Meyer (2013).

Apart from the work of the group above, the literature has a few more studies that demonstrate ways to identify threshold concepts in programming. However, some of these studies fail to provide empirical data to support their arguments, and as such, they are typically based on the researchers’ own perspectives on the subject. Nevertheless, to provide a complete picture of the literature, some of these works are concisely discussed in the following paragraphs.

One of the first studies in computer programming and threshold concepts is the work of Drummond and Jamieson (2005). Their research focuses on first-year novice programmers in an object-first approach to programming using BlueJ. To collect their data, they conducted interviews and use students’ learning logs. After the analysis of their results, the researchers indicate that the basic concept of a software, object, and class declaration are potential threshold concepts. Additionally, Khalife’s (2006) research paper explores the major problems that students encounter when they start programming and suggests a computer model that considers threshold concepts for basic programming. Specifically, he proposes the “concrete model of the computer internals and its operation during program execution” to be the first threshold concept a student confronts (Khalife, 2006: 246). He further recommends that curriculum design should first concentrate on a model of general
instructions that protects students from syntax specifics and further visualises the code execution before students get involved with the development of a program. It is indeed common knowledge in the community of programming that students experience difficulties regarding the notional and the physical machine (Sorva, 2013). The investigator further applied his model, and the initial results designate that there is a significant improvement in students’ academic achievement and perception of computer programming.

In addition, program – memory interaction is recommended as a potential threshold concept by Vagianou (2006). She supports her claims based on the fact that this concept appears to have some of the characteristics of threshold concepts. Once more, one of the limitations of this study is that the researcher does not support her arguments with empirical data; rather, she reflects upon her own experience. Another threshold concept, recursion, is also proposed in the study of Rountree and Rountree (2009) as the authors assert that it possesses all the characteristics of thresholds. An additional study of Sien and Chong (2011) regards the area of object-oriented modelling as a set of associated threshold concepts. They have proposed the concept of class, generalisation-specialisation hierarchies, and object interactions as candidate threshold concepts.

Finally, an alternative approach to identify threshold concepts in programming was proposed by Holloway, Alpay and Bull (2010). Specifically, they attempted to quantify students’ experience, admitting that previous qualitative efforts have proved to be unclear. They have selected to explore the concepts of “recursion, binary trees, memory allocation, object-orientation, and prolog backtracking”. The method of investigation was surveys designed to include questions addressing the threshold concepts’ characteristics. Students also completed scaled questions via a web page in which they also answer questions regarding
which concept was more difficult for them to understand and which one was more transformative for programming. After the statistical analysis, the authors concluded that recursion and object orientation are found to include the troublesome characteristic while they suggested that the integrative characteristic cannot really be captured with questionnaires.

To conclude this section, research in programming has identified concepts that can be regarded as thresholds, with the most important work so far being that of Eckerdal and her group of Computer Education researchers. The following table summarises the work that has been done so far and the results of these studies.

*Table 1.2 Summary of Threshold Concepts Identified in the Literature by chronological order—The star (*) denotes studies conducted by the same group of researchers*

<table>
<thead>
<tr>
<th>Researcher(s)</th>
<th>Threshold Concepts</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drummond &amp; Jamieson (2005)</td>
<td>Object, Class Declaration</td>
<td>Students’ Interviews</td>
</tr>
<tr>
<td>Eckerdal et al. (2006)*</td>
<td>Abstraction Orientation</td>
<td>Reflection on Personal Experiences and/or Literature</td>
</tr>
<tr>
<td>Khalife (2006)</td>
<td>Model of Computer’s internals and its operations during execution</td>
<td>Reflection on Personal Experiences and/or Literature</td>
</tr>
<tr>
<td>Vagianou (2006)</td>
<td>Memory Interaction</td>
<td>Reflection on Personal Experiences and/or Literature</td>
</tr>
<tr>
<td>Boustedt et al. (2007)*</td>
<td>Levels of abstraction, Pointer, Classes, Objects, Instances, Recursion, Induction, Procedural Abstraction, Polymorphism,</td>
<td>Instructors’/Students’ interviews</td>
</tr>
</tbody>
</table>
2.1.7 Concluding Remarks and Critique

The review of the literature has demonstrated a rich repertoire of methods employed for identifying threshold concepts. Nevertheless, the identification process is difficult and requires time, reflection and debate. Barradel (2013) suggests that it seems that in the first stages when a preliminary list of threshold concepts is constructed, consensus is significant. For this reason, she recommends the use of the Nominal Group Technique (NGT) and the Delphi Technique which are both examples of consensus approaches and appropriate for exploring both individuals’ and a group’s point of view.

Male and Ballie (2011: 252) argue that to identify threshold concepts the most appropriate source is to collect data either directly from students or from “people whose experiences give them awareness of students’ experiences”. They further argue that teachers can identify
not only concepts that are troublesome but concepts that are transformative for students. Additionally, Shinners and Kennedy (2013) point out that research in this field has reached a dead end. They criticise the methods used for identifying threshold concepts in programming, emphasising that asking students about difficulties they confronted in the past is an unreliable method. Instead, they advocate interviewing teachers to identify threshold concepts. They note that “there, after all, is where the reality of student learning is lodged, in the day-to-day classroom experience” (2013: 14). They also state that the identification of threshold concepts needs both pedagogical and content knowledge on behalf of the interviewees and they suggest that new research attempts should direct their focus on teachers’ pedagogical content knowledge to create teachers’ concept representations (CoRe’s). For this reason, Zwaneveld et al. (2016) contend that this can be found in secondary teachers rather than university teachers. They further advocate an approach that employs both teachers and students for this process.

Nevertheless, the theory also faces criticisms not only for the lack of a methodological framework but also for the constructs that it negotiates. For example, it has been claimed that the whole notion of threshold concepts is ambiguously described and vaguely explained (Rowbottom, 2007; Bradbeer, 2006).

Moreover, O’Donnell (2010) and Rowbotoom (2007) argue against the characteristics of threshold concepts as defined by Meyer and Land (2003). Particularly, they claim that these characteristics are weakly described (pointing to the “potentially” word that is used in the definition of them) making in this way any concept potentially a threshold one. For this reason, Quinlan et al. (2013) point out that the first step towards clarity is to illuminate the fundamental constructs of this theory. They suggest that a different identification approach
should be employed for each of the thresholds’ characteristics. For instance, if the troublesome characteristic is under investigation, then the researcher should focus on empirical studies in order to identify what students find difficult. Accordingly, if the researcher is concentrating on the integrative characteristic, then interviews, card sort methodology or focus groups are proposed as suitable approaches. Conclusively, each of the researchers’ attempts for identifying threshold concepts should focus on different questions, one for each of the characteristics (Quinlan et al., 2013).

Ultimately, Meyer and Land (2005) emphasise that the methods a researcher chooses to employ for the identification of threshold concepts should be suitable for discovering a variation in students’ experiences. However, even though a substantial amount of research studies has been conducted to identify threshold concepts in various disciplines, the question yet remains on how researchers should approach the identification problem.

2.2 Misconceptions

This section discusses the notion of misconceptions. It starts with locating misconceptions in constructivism and then continues by describing fragile knowledge and sources of misconceptions in computer programming. The section finishes with examples of students’ misconceptions in the most researched areas of computer programming.

2.2.1 Misconceptions in Learning

From a constructivist point of view, a child’s existing schemata determine what he/she learns through experience or instruction; but, if a child is not able to persevere with these processes, then he/she will be led to gaps in knowledge and misconceptions (Sarwadi & Shahrill, 2014). However, misconceptions and students’ errors are not to be seen as a bad and fearful situation
Regarding this issue, Papert (1980:23) reminds us of the importance of not being afraid to be wrong and make errors but instead, of concentrating on how to fix these errors:

"Many children are held back in their learning because they have a model of learning in which you have either ‘got it’ or ‘got it wrong.’ But when you program a computer you almost never get it right the first time. Learning to be a master programmer is learning to become highly skilled at isolating and correcting "bugs", the parts that keep the program for working. The question to ask about the program is not whether it is right or wrong, but if it is fixable. If this way of looking at intellectual products were generalized to how the larger culture thinks about knowledge and its acquisition, we might all be less intimidated by our fears of ‘being wrong.’" Papert (1980:23)

Sewell (2002) argues that misconceptions or incorrect views are part of prior knowledge and, as a result, their role is vital in learning. Along the same lines, Martin et al. (2002) describe misconceptions as ideas that provide a wrong understanding and are created by a person’s experience. As Smith et al. (1994) argue, students come in class with already formed conceptions about some of the scientific phenomena described during instruction; but these already shaped conceptions have different characteristics than those concepts presented in class. They further argue that these differences are known in the research literature by a variety of names such as “misconceptions, preconceptions, alternative conceptions, naive beliefs, alternative frameworks, and naive theories, or misunderstandings” (Smith et al., 1994: 119).

The problem is that once misconceptions are moulded in a child’s mind, it is enormously challenging to modify those (Eggen & Kauchak, 2004). This is because misconceptions tend
to persist. Indeed, Longfield (2009) advocates that it is easier to learn new information when it can be connected with what a person already knows. Nevertheless, he argues that humans, on the one hand, have the propensity to amplify information that aligns with their current theories, but on the other hand, they ignore information that will lead them to disequilibrium (Longfield, 2009). To achieve this, humans reject any new information that disagrees with their current views. The extent to which these current views and students’ distinctive version of knowledge are not aligned with scientific knowledge, determines the misconceptions students’ hold in their discipline (Ben-Ari, 2001) and this is the definition I also consider in this thesis.

Thus, in education, two questions are of significance regarding misconceptions: How can we identify them, and once we do how do we combat them? Literature about misconceptions, and about addressing them, profoundly concentrates in the sciences (e.g., Treagust, 1988; Doran, 1972; Thompson et al., 2006; Cakir, 2008); Specifically, in the “Science Teaching Reconsidered: A handbook” (NCR, 1997:28), five types of science misconceptions are listed: “Preconceived Notions, Nonscientific Beliefs, Conceptual Misunderstandings, Vernacular Misconceptions, and Factual Misconceptions”. Briefly, preconceived notions are deep-seated in everyday experience. Non-scientific beliefs are views gained by students from non-scientific sources like myths and religion. Conceptual misunderstandings are constructed “when students are taught scientific information in a way that does not provoke them to confront paradoxes and conflicts resulting from their preconceived notions and non-scientific beliefs” (p.28). Vernacular misconceptions are created from words “that mean one thing in everyday life and another in a scientific context” (p.28). Finally, factual
misconceptions arise from inaccuracies learned at a young age and maintained unrestricted until a child reaches adulthood.

In addition, literature on misconceptions also involves the disciplines of chemistry (Ozmen, 2004; Abraham et al., 1992; Kerr & Walz, 2007), biology (Boo, 2005; Simpson, 1988; Lazarowitz & Lieb, 2006), engineering (Streveler et al., 2003; Miller et al., 2011), physics (Brown, 1992; Clement, 1982), and computer science which will be further discussed in the following paragraphs.

2.2.2 Misconceptions in Computer Programming

Students’ achievement in computer programming depends on many factors such as their enthusiasm towards programming (Akinola & Nosiru, 2014), their self-efficacy (Askar & Davenport, 2009), and their cognitive and problem-solving skills (Sheth et al., 2016). Although several studies have concentrated on identifying these factors, it is difficult for teachers to use these findings to alter their pedagogical approach to teach programming.

Students in computer programming, and general, novice programmers, make lots of mistakes. Some of them centre on syntax errors, whereas others have deeper causes and stem from misconceptions or misunderstandings in programming. Undoubtedly, understanding students’ imprecise conceptualisation is extremely important and a prerequisite for assisting them towards a more precise conceptualisation (Kaczmarczyk et al., 2010). Especially in computer programming, such an attempt is critical, since if early misconceptions are not resolved, it will be impossible for students to carry on with this discipline due to its sequential nature. Therefore, many researchers, since the late 80s, centred their attention on the identification of these misconceptions in computer programming, with the ultimate goal
to assist teachers and academics in improving their courses and instruction and in reducing learning difficulties.

2.2.2.1 Fragile Knowledge and Sources of Misconceptions in Computer Programming

One of the earliest studies in this field is Perkin’s and Martin’s (1986) who investigated students’ difficulties in the BASIC programming language. By conducting clinical interviews, and by employing a constructivist approach, they were able to identify four types of fragile knowledge, “knowledge that is partial, hard to access, and often misused” (Perkin’s & Martin’s, 1986:4):

1. Partial knowledge: refers to knowledge that is missing due to students’ lack of retaining to memory or because the students never learn it.

2. Inert knowledge: refers to knowledge that a student has previously acquired, but he/she was not able to recall it.

3. Misplaced knowledge: refers to knowledge that in fact, is appropriate for some occasions other than the ones the student actually used it.

4. Conglomerated knowledge: refers to congregating pieces into a compressed group. Specifically, in computer programming, it is a mishmash of code that roughly demonstrates an anticipated plan and indicates students’ difficulties to bring together the fragments of code in an appropriate manner.

However, more concrete discoveries in the curriculum of computer programming were made by Du Boulay (1986) who identified specific areas that students find difficult in the curriculum, and he grouped these areas into categories. Specifically, according to Du Boulay (1986), the first factor of difficulty that students will confront in computer programming is...
the “Orientation”. It includes students’ understanding of the importance, the reasons why, and the advantages of using computer programming for addressing problems. Then, the next frame of difficulty for students is the “Notional Machine”. In a nutshell, notional machine refers to the underlying functions that the computer performs when the program is executed. Du Boulay (1986), and later Sorva (2013), emphasises the importance of students to understand these mechanisms but also the difficulty for them to accomplish this task. The third factor of difficulty is “Notation”, which is the phase that students need to grasp the syntax and semantics of the programming language under learning. Although Spohrer and Soloway (1986) argue that students’ errors/bugs in computer programming do not really stem from their misunderstandings of the language’s syntax, the latter is an expected initial difficulty that students need to confront.

Last but not least, students need to build “Structures”, which are plans that they can employ to create small-scale programs. The final category, according to Du Boulay (1986), is “Pragmatics”, which refers to the ability to specify, develop, test and debug a program. Du Boulay (1986) also underlines that none of the above categories must be considered separately from the others. In fact, the students deal with all these difficulties at once.

Most significantly, Du Boulay (1986) highlighted specific sources of misconceptions for students who learn computer programming. He categorised the type of students’ programming mistakes in the following three areas, Misapplication of analogy, Overgeneralisations and Interactions. Briefly, a misapplication of analogy denotes situations where there is overuse or overestimation of the strength of an analogy. For instance, in computer programming, it is a common teaching approach to present variables as a box. As helpful as may initially, this analogy seems, students may lead to the assumption
that variables can hold many values as boxes can contain many items. *Overgeneralising* refers to the general conventions that students and novices make that are incorrect, and usually these overgeneralisations refer to syntax. Finally, *Interaction* refers to students’ mistakes when they incorrectly incorporate sub-parts of the program.

Soon after Du Boulay’s (1986) work, Perkins and Simmons (1988) explored categories of misconceptions in science, math and computer programming. They identified and classified four levels of knowledge that result in misunderstandings, that is, Content frame, Problem-solving frame, Epistemic frame and Inquiry frame. The *Content* frame includes knowledge about the focal concepts of the discipline. Specifically, in computer programming, such concepts include the notions of variable, expression, assignment, statements, loops etc. As the following section would demonstrate, this is a vital frame because many misconceptions have been linked to these focal concepts. Specifically, the *Content* frame has three kinds of misunderstandings linked to it: *Naive*, *underdifferentiated*, and *malprioritised concepts*, which are concepts referring to preconceptions, students’ previously formed understanding of them. The second is the *Difficulties in accessing knowledge*. These refer to inert knowledge, the knowledge that students possess but initially cannot retrieve and last, *problems of garbled knowledge* which refer to mistakes occurring when students transfer the characteristics of one command into another. In more recent literature, this type of misconception was further enriched by Shmallo et al. (2012) by referring to students’ tendency to expand or reduce the features of a concept.

Perkins and Simmons (1988) emphasised that misconceptions hinge on inadequacies in the content frame. Still, they contend that misconceptions also are accumulated by deficiencies in the other frames. Particularly, the problem-solving frame includes general problem-
solving strategies as well as specific domain problem-solving knowledge and general heuristics (e.g., breaking down problems, different seeking paths). However, novices may stack with strategies that are not working in a specific situation. The following are examples of these strategies: Trial and error, Preservation and quitting, Proceeding on a guess, Stock responses, Equation (Perkins and Simmons, 1988). Finally, regarding the other frames, examples of sources of misunderstandings are intuitions that mask contrary observation (Epistemic Frame), intuitions have priority over internal coherence (Epistemic Frame), confirmation bias (Epistemic Frame), No problem finding (Inquiry Frame), Academic applications only (Inquiry Frame), and No venturing (Inquiry Frame) (Perkins and Simmons, 1988).

Other sources of misconceptions are also reported in the literature. Bonar and Soloway (1983) argue that students’ prior knowledge of natural language specification is a source of difficulties in programming, and even students’ prior knowledge in another programming language imposes difficulties when students endeavour to learn another language. Additionally, Spohrer and Soloway (1986) have noticed the influences of the English language to programming statements. This is because many of the computer programming command words are named after English language words; this relationship makes students confused about the syntax of programming statements.

However, a different source of misconceptions in programming was described by Shmallo et al. (2012). The researchers were not only interested in investigating misconceptions in object-oriented programming but in identifying if common characteristics exist in students’ misunderstandings. Indeed, they were able to identify that students lean towards “expanding or reducing” concepts’ definitions. Specifically, expansion refers to allocating further
characteristics to a term than it actually holds, whereas the opposite happens with reducing
where the students constrain the features of the concept. To be more specific, they recognise
two main patterns for each category: For Expansion, the first pattern is “many rather than
one”. A misconception that fits in this pattern is, for example, that static variables can be
accessed by each object separately, that is, the static variable belongs to the object, and thus,
static variables have many occurrences. This misconception, probably, is caused by students
thinking of static variables as instance variables. The second pattern is “expansion of
properties”. For example, students may erroneously believe that the private variables of an
object can be used by methods of other-outer classes or that the default constructor can be
invoked automatically or that a class’s method can be called without using an object. For
Reduction, the first pattern is “one rather than many”. An example of a misconception for
this pattern is that a constructor can be called only once and, thus, only one object can be
created per class. The second pattern is “reduction of properties”. An example to
demonstrate this misconception is that two objects cannot have the same property values.
Additionally, a class’s variable cannot have a public access type and when an object is
created is simply the allocation of a variable cell.

However, if someone is interested in diagnosing students’ deficits in programming, McGill
and Volet’s (1997) model is a promising framework. In particular, McGill and Volet (1997)
proposed a synthesised conceptual model by integrating the categories of knowledge arising
from literature in computer science education, namely, syntactic, conceptual and strategic,
with cognitive psychology (declarative, procedural, and conditional). In this way, they
created a five-part framework: declarative-syntactic, declarative-conceptual, procedural-
syntactic, procedural-conceptual, strategic/conditional knowledge. Briefly, the
1. *declarative-syntactic knowledge* is associated with the programming language’s syntax. For example, knowing that semicolon ends each statement in Pascal or Java, or understanding and knowing the difference in syntax of a procedure and function in Basic.

2. *declarative-conceptual knowledge* is the ability to describe what happens during the execution of the program. For example, tracing through a fragment of code and explain what this fragment does or explaining where the result of a function is returned to.

3. *procedural-syntactic knowledge* is the capability to be able to understand and apply syntax rules. For example, to write correct loops in a specific language or to open a file and read from it.

4. *procedural-conceptual knowledge* is the ability to design and match solutions to programming tasks. For example, to write a function to calculate the mean of three numbers.

5. *strategic/conditional knowledge* is the ability to take one's comprehension of syntax and semantics to "design, code, and test a program to solve a novel problem" (p. 284).

### 2.2.2.2 Examples of Misconceptions in Programming

**Variables and Assignment**

One of the main areas that the literature has shown a variety of students’ misconceptions is the notion of variables and assignment in programming. Most of the studies addressing misconceptions about variables support that incorrect analogies account for some of them
(e.g. variable as a box, as illustrated earlier). Other studies regard that students’ lack of problem-solving skills has a significant impact on students’ ability to program while others account algebra knowledge responsible for variables’ misconceptions. Indeed, it seems almost natural for a student to assume that variables in programming act the same as mathematics’ variables when he or she engages with programming for the first time. Unfortunately, this previous mathematical understanding of variables and their association with programming variables can prohibit students’ understandings and create misconceptions in this area. According to Plass (2015), the main sources of misconceptions about variables stem from Mathematics, Human Interaction, Container Analogy, and Semantics.

Specifically, mathematical expressions are regarded as a relation between variables and equations that need to be resolved. Influenced by this view of variables, students regard programming expressions as mathematical expressions. A misconception that illustrates this notation is that students perceive the assignment operator as making both sides of the expression equal and as such, variables can interchange sides (Plass, 2015). Students also erroneously believe that programming expressions are equations that are processed by the computer. On this matter, Du Boulay (1986) further had noticed that students regard expressions of assignment as equations that link together the two variables (left and right variable of the equation); thus, when the value of one variable changes, it influences the value of the other. These misconceptions validate Perkins and Simmon’s (1988) categories of sources of misconceptions and specifically ‘preconceptions’ and ‘garbled knowledge’. Indeed, it is evident that students’ previous mathematics knowledge can lead them to erroneous assumptions for the use of assignment and variables in computer programming.
Ma et al. (2007a) also corroborate these misconceptions about assignment statements, and they further added that students perceive assignment as a comparator. Nonetheless, these findings highlight the importance of helping students distinguish the differences between mathematics and programming variables at introductory courses.

Human interaction also influences the way novice programmers write code (Spohrer & Soloway, 1986). Particularly, novice programmers, influenced by the humans’ ability to understand each other even with imprecise sentences, expect almost the same ability from the computer (Pea et al., 1987). In other words, students expect from computers to be able to understand their inaccurate actions. A common misconception that stems from this assumption is that the computer expects the values of a variable to be logical (Du Boulay, 1986). For example, a variable with the name “taller” cannot hold the shortest value.

In the container analogy, usually, the misconceptions stem from analogies made to help students understand an aspect of programming. Du Boulay’s (1986) was among the first to highlight this source of misconception in computer programming. Examples of such misconceptions include that a variable can contain many values and that initialising a variable to zero to calculate a sum is not needed (both of these misconceptions stem from the box-variable analogy, Du Boulay, 1986).

Object-Oriented Programming

Another well-studied area around students’ misconceptions is object-oriented programming. It is commonly accepted among programmers that it is quite difficult for someone who traditionally uses an imperative paradigm to adopt this paradigm and novices, particularly, experience this transition with many difficulties.
Students’ misconceptions about object-oriented programming were studied by many researchers like Fleury (2000), Chen et al. (2012), Shmallo et al. (2012), Holland et al. (1997), and Ragonis and Ben-Ari (2005). Most of these studies usually refer to the conceptual challenge of object-oriented programming concepts (e.g. classes, objects, static, etc.). A good illustration of such a study is Chen’s et al. (2012) which suggests a variety of object-oriented programming’s conceptual misunderstandings. Indeed, their study highlights students’ difficulties in understanding the differences between static data members and constant data members as well as the differences between classes and objects (e.g. students seem not to understand that classes are used to create objects). Holland et al. (1997) have previously noticed the confusion that the notions of objects induce on students. They raised the alarm when they argued that students often regard objects as a kind of a variable and they further explained that instructors should avoid creating classes with only one instance variable, just to avoid the creation of this misconception.

Additionally, they suggested that students usually regard objects as a repository due to their tutors’ examples that exaggerate the data feature of objects at the cost of their behavioural feature. These notations of Holland et al. (1997) are linked to Du Boulay’s (1986) sources of misconceptions and, particularly, overgeneralisation and Shmallo et al.’s (2012) expansion rules. Students seem to broaden the characteristics of a concept more than it is sensible; as a result, they create programming rules that lead to incorrect programs. Finally, students also have difficulties regarding memory allocation and objects. Specifically, Kaczmarczyk et al. (2010) state that students believe that objects are allocated the same amount of memory or even that objects that are not instantiated are allocated space in memory.
Ragonis and Ben-Ari (2005) have also contributed to this research area. They discovered students’ tendency to regard that an object’s attribute value can be used as an identifier for an object. This is almost in line with Holland’s et al.’s (1997) observation that students seem to be confused when comparing objects because they tend to regard objects with same name values as the same objects. On the other hand, but within the same area, other students seem to believe that two objects of the same class cannot have the same property values (Ragonis & Ben-Ari, 2005).

Another blurry area for students in object-oriented programming is constructors. Constructors are used in this programming paradigm to create an object and allocate an appropriate amount of memory but also to initialise the objects’ properties. This whole notion of constructors and their role are an extra conceptual difficulty for novice programmers. To be more precise, the three main misconceptions about constructors are that they are private methods since they belong to the class, that default constructors are invoked automatically when the program starts, and finally that constructors need return types since they resemble methods (Chen et al., 2012). These notations are perfect examples of Misplaced knowledge as identified by Perkin’s and Martin’s (1986) and problems of Garbled knowledge defined by Perkins and Simmons (1988); students use their experience of one occasion (methods) to another (constructors), and they transfer characteristics of one concept to another.

Furthermore, Ma (2007) also mentions that students believe that objects are created automatically and, thus, they often neglect to create constructors. In the same grounds, Ragonis and Ben-Ari (2005) discovered that students believe that constructors are only comprised of assignment statements for the instantiation of the class properties, an argument
also mentioned by Fleury (2000) who posits that students regard that the constructor’s role is to initialise the object’s variables. This is of course, again, in agreement with Holland et al.’s (1997) explanation of the sources of these misconceptions, which is mostly attributed to the instructors’ early illustration examples of object-oriented programming. The first examples that a tutor demonstrates in class are constructors that just initiate the classes’ variables. Consequently, students are led to these misconceptions quite easily.

One more area in which students experience difficulties is arguments and parameter passing for method calls, which are of considerable significance to this study. Particularly, students find it difficult and often are confused between the use of actual and formal parameters (Fleury, 1991). Furthermore, Fleury (2000) argues that students often believe that actual parameters should only be of some numeric type to replace the formal ones. Sirkiä (2012) has also discovered an additional misconception in this field. She noticed that some students are confused with the order between the evaluation of parameters and function call. Furthermore, they also seemed to confuse the return statements of a function and where these should be located.

Regarding methods, Ragonis and Ben-Ari (2005) noticed that students hold the misconception that an object can call a method only once and that methods can be used to create an object. Ragonis and Ben-Ari (2005) further discovered that students hold the impression that the order of the methods’ execution depends on their order in the class. The same conclusion was also reached by Sirkia (2012) and Sleeman et al. (1989) who discover that students hold the misconception that the program executes the methods when it starts. These, subsequently, can be attributed to students’ misunderstandings of the way and the sequence a program executes its commands.
Efficiency of Algorithms

In computer science education, the students are presented with a dual challenge: On the one hand, the computer science students should learn and develop effective programs – programs that produce the expected result - and on the other hand, they should write efficient programs – performing at a minimum cost regarding both computation time and memory space. Efficiency is deeply ingrained in computational complexity, and its conceptual difficulty offers a fertile landscape for misconceptions to grow. Indeed, there are studies that investigated students’ understanding of algorithms’ efficiency, and they found that students’ perception of efficiency is not well-suited with the corresponding meaning in computer science.

Gal-Ezer and Zur (2004:6) point out four misconceptions students often hold in this area: “a) a shorter program is more efficient b) two programs containing the same statements are equally efficient (even if the order of the statements is different) c) the fewer variables there are, the more efficient the program” and d) programs achieving the same task have equal efficiency. The first misconception, according to the researchers, is rooted in the intuitive rule “more of A more of B” and the second and the last one follows the rule of “same A same B”. These are rules stemming from Science and Mathematics research on misconceptions (Stavy and Tirosh’s, 1996a, 1996b). However, it is clear that these intuitive rules resemble the epistemic frame, mentioned earlier, and the sources of misconceptions that it incorporates as defined by Perkins and Simmons (1988). Nevertheless, regarding the misconception about variables and efficiency, this leads to students’ misunderstandings about memory and how computers allocate and store variables, already discussed and confirmed by previous studies. A more recent study by Shah et al. (2015) agrees with the
previous findings and, validates these misconceptions but further complements that students also have misconceptions regarding loops and efficiency. In fact, Gal-Ezer and Zur (2004) discovered that students regard the same kind of iteration as equally efficient regarding the number that inner statements are executed which is also supported by Ozdener (2008).

2.2.3 Concluding remarks

In summary, this section presented a detailed description of the difficulties that computer programming concepts entail for students as well as their sources. From the very early programming concepts, like variables and operators, to more advanced concepts, like classes, objects and efficiency, students are confronted with misconceptions, which are attributed to different aspects. The evidence presented thus far, supports the idea of interdependence between misconceptions held in one concept and another, a fact that suggests that misconceptions in programming must be addressed as early as possible. Consequently, the burden again is upon teachers; it is essential for teachers to realise and be aware of misconceptions that the literature has revealed and to alter their teaching approaches respectively, to better assist their students. Sorva’s (2018) chapter “Misconceptions at the Beginner Programmer” provides a list of students’ misconceptions in programming as well as guidelines for misconception-sensitive teaching.

2.3 Affective dimensions of learning

This section aims to present the affective dimensions of learning that are of interest in this thesis: self-efficacy, calibration, task value, motivation, sense of belonging and identity. Research studies that link these constructs with threshold concepts have yet to be conducted to my knowledge. This was the purpose of studying these concepts in relation to threshold
concepts and, thus, to try to address this gap. Therefore, the section first attempts to define affect and then continues to present the above affective dimensions of learning and their role in education and specifically in motivation and performance. As enough studies in computer science and programming focus on students’ self-efficacy, there is a specific subsection that discusses these studies. On the contrary, studies on task value, sense of belonging and identity are rare in computer programming and, thus, are presented all together in the last subsection.

2.3.1 Defining affect

The role of affect and emotion in education has long been highlighted by many researchers. Among these, Hascher (2010: 13) highlights their importance by saying that “there is rarely any learning process without emotions”, while Jackson (2015) advocates affect as a central dimension of learning which influences, among others, students’ learning approaches, their levels of engagement, communication and collaboration with peers and teachers, their achievement and performance.

Defining affect and emotion is not a straightforward process, as conceptualisations and the way these are examined fluctuate within the disciplines (Linnenbrink, 2006 cited in Jackson, 2015). For Wetherell (2014), emotions are “the conventional cultural packets or prototypes for affect, e.g. anger, joy, sadness, disgust, shame, surprise” that “register evaluations of events, standpoints on what is happening, and investments” (Wetherell, 2014 cited in Illeris, 2018). Affect, she argues, is a “broader, more generic term. It includes reactions that may be difficult to categorise, and which may not be organised into conventional categories” (Wetherell, 2014 cited in Illeris, 2018). In educational settings, Pekrun and Linnenbrink-
Garcia (2014: 2-3) point out that affect represents a broad category of non-cognitive constructs (which contain emotion) and it also signifies beliefs, motivation and self-concept. On the contrary, Breckler and Wiggins (1989) locate thoughts, beliefs and judgments as constructs of evaluation – which for them refer to cognition - and affect then refers to manifested emotions and feelings stimulated by an attitude object. In psychology, an attitude object is “any distinct object, such as a target, a person, an event, an abstract entity, a social group, the self, or any aspect of the world, that is evaluated” (McCulloch and Albarracin, 2009:60).

According to McLeod (1986;1989;1992), there are three basic descriptors of the affective domain: beliefs, attitudes and emotions while De Bellis and Goldin (2006) include a fourth category that of values/morals/ethics. In the same line, Grootenboer and Marshman (2016) regard affect as a group of beliefs, values, attitudes and emotions, while recognising that motivation and engagement can also be included. The following figure (figure 2), adapted by Grootenboer (2003), highlights that beliefs, values and attitudes are seen as interconnected in the literature whereas feelings and emotions are seen as discrete from beliefs and values but associated with attitudes. Additionally, the ranges on the intensity of cognition and affect are depicted at the bottom of the figure.
2.3.2 Motivation in education

One of the examined constructs in this thesis is motivation. Motivation as a construct is usually used to describe human behaviour. It is highly linked with humans’ reactions, behaviours and fulfilment of needs as motivation provides reasons to individuals to act.

Brophy (1998) refers to motivation as an internal state or condition that initiates action and directs behaviour. In agreement, Weiner (1992) postulates that motivation is the desire of a person to act under specific ways while Keller’s (1991) definition highlights the degree of effort that individuals employ towards specific tasks.

In education, motivation plays a significant role, and that is evident in the numerous research studies around students’ motives in learning that cover a variety of subject fields. Motivation in education settings refers to students’ experiences and willingness to participate in school and learning activities, but also it refers to the incentives that drive students into participation (Green & Sulbaran, 2006). Gopalan et al. (2017) further add to this definition by saying that motivation offers positivism to students to achieve a task no matter the difficulties.
There are several motivation theories, yet one of the most known is Intrinsic and Extrinsic motivation theory. Intrinsic motivation refers to the individual’s participation in activities for personal contentment, enjoyment, curiosity and satisfaction that stem directly from the act of participation without any external reward anticipation (Deci & Ryan, 1985). In contrast, extrinsic motivation refers to behaviours and actions initiated by reasons other than an individual’s contentment and usually refers to activities such as rewards, benefits and gains or punishments. This particular kind of motivation, even though it is accompanied with high levels of engagement and willingness, tends not to withstand long and reinforces students’ behaviour to act or participate in activities only to gain rewards and not for personal development or enjoyment (Gopalan et al., 2017). On the other hand, students with intrinsic motivation tend to search for opportunities to gain knowledge and skills for personal growth and not for gaining high marks or social recognition (Ryan & Deci, 2000). However, Li and Lynch (2016) point out that both intrinsic and extrinsic motivation is necessary for the learning process.

Two of the motivational constructs that can accurately forecast students’ motivation are self-efficacy beliefs and task value (Bong, 2001). The following two subsections discuss self-efficacy, calibration, and task value - three of the examined constructs in this thesis - and present some studies contacted in this area regarding computer programming.

2.3.3 Self-efficacy and calibration

Beliefs, Gilbert (1991) state, have a cognitive structure that assists in the organisation of the information received, and in constructing an understanding of the world, thinking and reality. However, beliefs about oneself, convey a strong affective load regarding one’s confidence,
self-image, success or failure (Blano et al., 2010). Of particular importance in education research are beliefs of self-efficacy which impact individuals’ actions and tasks, the level of effort and their determination to pursue and continue on activities and tasks regardless the difficulties as well as the amount of stress they experience, their expectations about the outcomes and the self-regulation process. Bandura (1997:214) emphasises the significance of self-efficacy beliefs when he states, “the major goal of formal education should be to equip students with the intellectual tools, efficacy beliefs, and intrinsic interests needed to educate themselves in a variety of pursuits throughout their lifetime”.

The social cognitive theory recognises the role of self, environment and cognitive processes and their interactions in comprehending and discovering human behaviour. Bandura’s social cognitive learning theory places self-efficacy in the centre of learning and represents a person’s belief and judgements of her capabilities to accomplish a specific task (Bandura, 1977a; 1997b). Specifically, Bandura (1994) defines self-efficacy as "people's beliefs about their capabilities to produce designated levels of performance that exercise influence over events that affect their lives" (Bandura, 1994:2). As such, self-efficacy is a fundamental influential belief about a person’s capabilities, and it does not concern with skills, but with the perception of what can be accomplished with these skills.

Bandura (1994) suggested that students' self-efficacy can be formed by the selection and interpretation of information from the following sources: mastery experience, vicarious experience, verbal persuasion and physiological states. Mastery experiences are the most dominant source of efficacy as they are grounded in individual experiences and their results. These experiences could, for example, occur when someone attempts to accomplish something and succeeds. Because people believe in their ability to do something new when
they have a successful experience with something similar, these mastery experiences are the most efficient means to enhance self-efficacy (Bandura, 1994). Vicarious experience stems from observations and comparisons of the outcomes obtained by a role-model. For example, by observing someone like oneself succeeding on doing something one would like to do, boosts self-efficacy while the opposite (watching someone fail) has the reverse effect. Bandura (1994) states that vicarious experience depends on the degree of how close an individual thinks his/her model is to himself/herself. This means that the more one connects with the person being observed, the bigger the effect on her/his belief of accomplishment would be. Verbal persuasion also influences the efficacy and highly depends on the integrity of the person involved, while physiological reactions impact self-efficacy as emotions often influence a person’s perception of his/her capabilities (Zimmerman, 2000). Tense conditions evoke intense emotions, which impacts an individual’s perceived efficacy on a task (Bandura and Adams, 1977). For instance, Pajares (2002) points out that feelings of fear or anxiety contribute to a self-fulfilling prophecy of the inability to do a task.

2.3.3.1 Self-efficacy and achievement

Many years of research have highlighted the importance of self-efficacy beliefs on students and in general on a person’s cognitive and non-cognitive functions. Research evidence supports that students with high self-efficacy work harder, have higher persistence while demonstrating fewer emotional responses than students with low self-efficacy (Zimmerman, 2000). Additionally, self-efficacious students engage more with challenging tasks than other students while they demonstrate better performance (Bandura & Schunk, 1981; Zimmerman & Kitsantas, 1997, 1999; Schunk et al., 1987, Salomon, 1984, Schunk, 1981; Multon et al., 1991). Specifically, self-efficacy can affect performance in two ways: the first refers to a
direct influence and the second includes an indirect way by affecting other constructs like personal goals and commitment (Latham & Locke, 1991). Furthermore, when the goal level is controlled, then the relationship between performance and self-efficacy is linear (Locke & Latham, 1990; Bandura, 1989).

Despite self-efficacy’s influence on motivation and performance, Zimmerman (2000) advocates its impact on students’ self-regulation learning. He argues that self-efficacious students stimulate their learning by using self-regulatory strategies and processes such as “goal-setting, self-monitoring, self-evaluation and strategy use” (Zimmerman, 2000:87). Previous studies illustrate that self-efficacious students set and engage in more challenging goals and when the above constructs are used together, they are highly predictive of students’ performance (Zimmerman et al., 1992; Zimmerman & Bandura, 1994).

Erel (2000) has also emphasised the role of self-efficacy and its effect on self-regulation. She argues that self-efficacy affects goal setting, a subfunction of self-regulation, and as such, efficacious individuals set higher goals and are decisively dedicated to them. Specifically, differences between an individual’s goals and performance lead to self-dissatisfaction which in turn impact self-efficacy and may result in goal abandonment (Locke & Latham, 1990; Early & Lituchy, 1991; Bandura & Cervone, 1983 cited in Erel, 2000). Thus, many researchers advocate that self-efficacy mediates the relation between personal goals and performance (e.g. Early & Lituchy, 1991, Latham & Locke, 1991).

Additionally, other studies demonstrate that students with high self-efficacy monitor their work time much better than inefficacious students (Bouffard-Bouchard et al., 1991) while also employing learning strategies and self-evaluation processes to promote learning and
assess the results of their self-monitoring correspondingly (Zimmerman & Bandura, 1994; Zimmerman & Martinez-Pons, 1990).

2.3.3.2 Self-evaluation and Calibration

Self-evaluation is one of the central processes in self-regulation, and it refers to the individual’s evaluation of his/her performance (Ramdass & Zimmerman, 2008). These judgments must be as accurate as possible in order for the self-regulated process to be effective (Zimmerman, 1998). When students’ perceived performance aligns with their actual performance, then students are considered as well-calibrated. As a meta-cognitive judgment of an individual’s perceived performance and the actual one on a task, calibration has a critical role in learning (Ramdass & Zimmerman, 2008).

Measures of calibration that are usually used in research, and have also been employed in this research, are the bias or "direction of the errors in judgment" and it is calculated by subtracting the actual performance from the students’ self-evaluation score (Pajares & Miller, 1997; Ramdass & Zimmerman, 2008:29). A zero bias indicates perfect calibration, a positive suggests overconfidence and a negative under-confidence (Ramdass & Zimmerman, 2008). The second measure is the calibration accuracy which is the extent of the judgment error, and it is calculated by subtracting the absolute value of the bias score from its maximum possible value (Ramdass & Zimmerman, 2008). For instance, the calibration accuracy for a self-evaluation Likert scale (0 to 10) will range from 0 suggesting a complete inaccuracy to 10, suggesting complete accuracy.

Accurate calibration is focal for functioning, and it is important in metacognition and, therefore, in self-regulation (Sheldrake et al., 2014). Research studies suggest that students
who overvalue their capabilities and feel overconfident engage with difficult tasks which may lead to failures and therefore, to a decrease in their motivation (Kolovelonis et al., 2013). On the other hand, students who undervalue their capabilities and are under-confident are not engaged with challenging tasks, and this may lead to inadequate opportunities for development (Kolovelonis et al., 2013). However, calibration has also implications on attainment and school subject choices. For example, students who are overconfident with their skills may select subjects or further education studies that are beyond their capabilities and can lead to dropouts whereas under-confident students avoid joining subjects or further education departments that they may otherwise enjoy and succeed in.

Both self-efficacy beliefs and students’ calibration are in the centre of human functioning. Effective functioning entails skills and self-efficacy beliefs to perform a task and as such skills and knowledge alone are not sufficient (Artino, 2006). Bandura’s self-efficacy component of his theory has a deep effect on research on motivation and achievement in education settings. It is regarded as a dynamic construct which changes over time and as such, it is studied as something that can be changed with instructional interferences. In the following section, I discuss research studies in the field of computer science education and specifically in programming and students’ self-efficacy.

2.3.3.3 Self-efficacy in computer programming courses

As in many other fields, the investigation of the connection between students’ self-efficacy in programming and their success or performance in this course has been the focus of some research studies. Based on relative educational research in other fields, which recognise the intertwined relationship between self-efficacy beliefs and skills, most of the research conducted in computer programming courses seeks to explore and establish a relationship
between self-efficacy and academic success or performance in computer programming rather than investigating interventions that can boost students’ self-efficacy in this course.

An example of such research is the study of Wilson and Shrock (2001) who investigated factors that contribute to success in an introductory computer science course. The factors that the researchers were interested in were “math background, attribution to luck for success/failure, domain self-efficacy, encouragement, comfort level, work style preference, previous programming experience, previous computer experience and gender”. The participants were 105 college students and the measures employed included the Computer Programming Self-Efficacy Scale developed by Ramalingam and Wiedenbeck (1998) and a questionnaire about students’ gender, math, programming and computer previous experience, encouragement, comfort level, and other factors of interest. The results of the study indicate the importance of three predictors: comfort level, maths, and attribution to luck for success/failure (negative influence) with the comfort level in the computer science class being the best predictor for success. Interestingly, the authors do not discuss any connections between self-efficacy and success.

A more detailed study, with a focus on self-efficacy, was conducted by Wiedenbeck et al. (2004). Their interest concentrates more on the relationship between students’ self-efficacy, their mental models in programming and performance. Using a path analytic approach, the researchers explore a model of factors that affect performance in programming and specifically the factors of “self-efficacy, mental model and previous experience” (pp.1) effects on learning programming. In total, ninety undergraduate students (2nd and 3rd year) took part in the study with most of them coming from majors other than computer science. The instruments included a background questionnaire (e.g. computer and programming
experience), a self-efficacy questionnaire and specifically the Computer Programming Self-Efficacy Scale (CPSES), developed by Ramalingam and Wiedenbeck (1998), and two instruments for mental models, that is program comprehension and program recall. The program comprehension included six short C++ programs and a list of five true or false questions covering Pennington’s categories of mental models (elementary operations, control flow, data flow, program function and program state) and program recall task which was used to measure mental organisation/models. The study took part in the second and thirteenth week of the semester. The results suggest that previous experience in programming and students’ mental models affect self-efficacy, which increased during the course and impacted success in the course.

On the contrary, Davidson et al.’s (2010) study does not support that self-efficacy is increased during a course. Specifically, the findings suggest that there was not any significant difference in students’ self-efficacy scores at the beginning of the introductory programming course and the end. However, the design of their research is quite questionable as the two self-efficacy scores refer two different groups – the first group was just starting the introductory programming course and the second had taken the course the previous year.

By employing the same methodology and instruments with the Wiedenbeck et al. (2004), Ramalingam et al. (2004) sought to explore the effects of self-efficacy and students’ mental models of programming on learning to program. The research participants were 75 university students who were enrolled in a computer science course, and some of them were majoring in computer science. The study was conducted two times: the first one was conducted in the second week of the semester in which the students were administered a background information questionnaire and the self-efficacy scale (CPSES, pre-scale), and the other in
the thirteenth (total fifteen) week in which the students completed the same self-efficacy scale and two tasks for the investigation of their mental models. The results of their study suggest that students’ self-efficacy increased significantly during the course, with the highest increase in students with low self-efficacy (pre-scale). There were also students that their self-efficacy decreased, which the researchers explained as an overestimation of their self-efficacy at the beginning of the course. The researchers concluded that previous experience strongly predicts students’ self-efficacy (pre and post), that a strong mental model increases self-efficacy and that internal mental models and self-efficacy impact students’ performance.

In 2005, Wiedenbeck developed and evaluated again a model of factors that influence students’ learning in programming, including again self-efficacy, previous experience in programming but this time instead of mental models (investigated in 2004) the researcher included the “knowledge organisation”. The participants of the study were 120 non-majors, and the measurements employed were a background questionnaire, a self-efficacy scale and a recall measurement. The data were collected in two phases: the first was between weeks the first and the eighth week of the start of the course and the other between the ninth and sixteenth week. The results of the study indicate that self-efficacy increased significantly, that previous experience is a strong predictor of self-efficacy (pre) but not knowledge organisation, and that both self-efficacy(post) and knowledge organisation affect performance in the course. Some of the students also experience overconfidence in programming.

In another similar study, Giannakos, Hubwieser and Ruf (2012) pointed out an important difference between high school and undergraduate students’ self-efficacy in programming. Specifically, the researchers conducted a study to investigate students’ perceptions in
computer science, to compare students’ perceptions among German and Greek high school students and to explore potential differences in students’ perceptions in computer science at the end of secondary school and the start of undergraduate studies. The researchers employed a two-part questionnaire: the first part had questions about participants’ gender and age and the other included scales on different factors that the literature identifies as influential like, “performance expectancy, satisfaction, social influence, self-efficacy, behavioural intention, problem-solving confidence, confidence for using data commands, and confidence for data structure”. In total, 115 students participated in the study. The most exciting finding of their research is the significant difference between the undergraduate students’ self-efficacy and high school students with the latter appearing more efficacious than the former. The researchers justify this finding by arguing that freshmen students confront far more compelling problems which may affect their perceived capabilities.

By including the gender factor, Askar and Davenport (2009) extend the previous studies and findings and examine the factors that may affect self-efficacy in programming and specifically in Java for first-year engineering students. Specifically, the researchers were interested in exploring gender differences and how previous experience with computing and family familiarity with computers affect their self-efficacy. The researchers develop a seven-point Likert scale about self-efficacy (JPSES) based on the Ramalingam and Wiedenbeck (1998) scale, which consists of 32 items (α=.99) and also included demographic data questions, family related question and questions about their computer experience. The questionnaires were administered at the beginning of the programming course to 326 engineering and science students. The results of their study showed a statistically significant difference between the self-efficacy scores of male and female students with the latter being
lower than the former. The researchers also found that the self-efficacy scores of computer engineering students were significantly higher than students from other engineering departments and that the self-efficacy was highly correlated with students’ prior experience with computers. What is even more interesting in their findings is the significant relationship between family members computer use, and specifically mother’s use, and students’ self-efficacy scores. On the contrary, a later study conducted by Kormkmaz and Althun (2014) on 378 undergraduate engineering students, do not support Askar and Davenport’s (2009) findings regarding gender differences which means that their study revealed no significant difference between the self-efficacy levels of the two genders.

In contrast to Askar’s and Davenport’s (2009) findings, Jegede’s (2009) research did not reveal any relationship between self-efficacy and previous experience in computing. In his study, the researcher specifically aimed at examining the self-efficacy in programming and its relationship with students’ past programming experience. In total, 192 engineering students participated in the study and the measurements employed were a questionnaire about students prior programming experience and a scale for students’ self-efficacy (JPSES). The findings suggest that the number of programming courses offered and their performance in these (weighted scores) significantly predicted students’ self-efficacy in programming which is in line with Wiedenbeck’s (2004) and Ramalingan et al.’s (2004) findings. However, his study failed to support the hypothesis that there is a significant relationship between self-efficacy in Java programming with students’ prior experiences in computing or the number of years’ experience in programming which is in contrast to the findings reported by Askar and Davenport (2009). Jegede (2009) provides justification for this, which is based basically on students’ personal interests and the organisation of the curriculum.
In the same vein, Owolabi and Adegoke (2014) explored the influence of intrinsic factors (gender, computer ownership, mathematics, computer experience) and institutional type on undergraduate students’ self-efficacy in programming (Java). The participants were 254 undergraduates from four different universities, and the measurements employed were the Computer experience scale (r=.84) and the JPSES for self-efficacy. Among their findings, self-efficacy in programming has a positive and significant correlation with mathematics and computer experience, contradicting Jegede’s (2009) findings, but only mathematics background and not computer experience contributes significantly to the variation in self-efficacy which contradicts the findings of Askar and Davenport (2009). They also found that 99.0% of the variation in self-efficacy is explained by changes in an institution which suggests the critical role of institutions in students’ confidence to program.

A final study on students’ self-efficacy was conducted by Kanaparan et al. (2017) who explored the relationship between programming self-efficacy, emotional engagements (enjoyment, interest and gratification) and performance in programming. The researchers employ questionnaires which were administered to 433 students in the last week of a programming course. Among the research instruments used, was the self-efficacy scale of Ramalingam and Wiedenbeck (1998), and the academic self-efficacy scale (Bresó et al., 2011). Their findings suggest a strong connection between self-efficacy and emotion engagement in programming, between enjoyment and performance, but negative relationships between interest, gratification and performance.

The following table summarises the studies above reporting the authors, the instrument employed to measure self-efficacy and their most important findings (regarding self-efficacy).
Table 2 Summary of findings of self-efficacy and programming

<table>
<thead>
<tr>
<th>Authors</th>
<th>Scales</th>
<th>Findings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wilson and Shrock</td>
<td>CPSES</td>
<td>✓ comfort level, maths, and attribution to luck for success/failure predict success. Self-efficacy was not correlated with success.</td>
</tr>
<tr>
<td>(2001)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wiedenbeck et al.</td>
<td>CPSES</td>
<td>✓ previous experience in programming and students’ mental models affect self-efficacy which increases during the course and impacts success in the course</td>
</tr>
<tr>
<td>(2004)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ramalingam et al.</td>
<td>CPSES</td>
<td>✓ previous experience strongly predicts students’ self-efficacy</td>
</tr>
<tr>
<td>(2004)</td>
<td></td>
<td>✓ strong mental model increases self-efficacy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ internal mental model of students’ knowledge and self-efficacy impact their performance</td>
</tr>
<tr>
<td>Wiedenbeck 2005</td>
<td>CPSES</td>
<td>✓ self-efficacy increases significantly during the course,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ previous experience is a strong predictor of self-efficacy (pre) but not knowledge organisation,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ both self-efficacy(post) and knowledge organisation affect performance in the course.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Students also experience overconfidence in programming.</td>
</tr>
<tr>
<td>Study Authors</td>
<td>Journal</td>
<td>Findings</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>Giannakos, Hubwieser and Ruf (2012)</td>
<td>✓</td>
<td>high school students more efficacious that undergraduate</td>
</tr>
<tr>
<td>Askar and Davenport (2009)</td>
<td>JPSES</td>
<td>✓</td>
</tr>
<tr>
<td>Jegede (2009)</td>
<td>JPSES</td>
<td>✓</td>
</tr>
<tr>
<td>Owolabi and Adegoke (2014)</td>
<td>JPSES</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>only mathematics background and not computer experience contributes significantly to the variation in self-efficacy</td>
<td>99.0% of the variation in self-efficacy is explained by changes in an institution which suggests the important role of institutions in students’ confidence to program</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Kanaparan et al.</td>
<td>CPSES</td>
<td><strong>strong connection between self-efficacy and emotion engagement in programming, between enjoyment and performance, but negative relationships between interest (which they justified to participants perceptions and their measurement), gratification and performance</strong></td>
</tr>
<tr>
<td>(2017)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>self-efficacy levels were medium,</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>no significant gender differences</strong></td>
</tr>
<tr>
<td>Kormkmaz and Altun</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2014)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Taking everything into consideration, it can be argued that the most cited instrument for measuring self-efficacy in programming is the one developed by Ramalingam and Wiedenbeck (1998) for C++, but it was also adapted in Java language. A more recent instrument was created by Danielsiek et al. (2018), which again was based on Ramalingam’s and Wiedenbeck’s work, and it is currently under investigation to enhance its validity.

The studies presented in this subsection offer contradictory findings with some research studies supporting the impact of self-efficacy on students’ performance and the impact of previous experiences on students’ self-efficacy in programming while others fail to support
these claims. In any case, the research conducted so far highlights the importance of conducting more robust research studies in this area and of developing more appropriate tools to measure factors that influence self-efficacy.

The following section introduces the other constructs of interest in this study: task-value, sense of belonging and identity.

2.3.4 Task value, sense of belonging and identity

Many researchers argue that a person’s choice, persistence, and achievement on a task can be explained by their ability beliefs (e.g. self-efficacy) but also by the degree to which they value the task (Wigfield & Eccles, 1992; Liem et al., 2008). Research studies suggest that students with positive task value employ more profound cognitive and metacognitive strategies (e.g. McWhaw & Abrami, 2001; Pintrich, 1989;1999; cited in Neuville et al., 2007). Specifically, the role of task value on students’ motivation and achievement is highlighted by Eccles et al. (1983). They suggested an expectancy-value model of achievement motivation to explore students’ decisions to dedicate time and effort to achieve a task (e.g. learning something) (Green et al., 2017). They postulate that expectancies and values are affected by task-specific variables such as “self-efficacy, perceived difficulty, goals, and affective memories” which they have an impact on “achievement choices, performance, persistence and effort” (Wigfield & Eccles, 2000:69). In their model, they recognise a variety of achievement values constructs: attainment value of importance, which refers to the “importance of doing well on the task” (p.280), intrinsic value, which refers to the satisfaction and pleasure of doing the task, utility value or usefulness (extrinsic), which refers to the degree to which a task is related to a person’s future goals, and cost, which refers
to the effort (including emotional effort) needed to do the task and to the level of which the decision to participate in a task affects other activities (Wigfield & Eccles, 2000:72).

Eccles et al.’s model assumes a positive relationship between expectancies and values. For example, both Bandura’s self-efficacy theory (1997) and Eccles et al. model posit that beliefs about one’s ability affect task values. On this issue, Wigfield (1994) postulates that initially, this relationship is possible to be independent. However, over time, children start to assign more value to tasks in which they perform well. This is because, performing well in a task results in a positive effect that is then attached to the activity performed or conversely, assigning a lower value to challenging tasks is a possible way to maintain a high self-efficacy (Eccles et al., 1983). Therefore, over time, students’ beliefs of confidence in achieving a task (self-efficacy) and students’ beliefs of worth pursuing a task (task value) are often positively correlated (Bong, 2001) with each other and with students’ performance.

The difference between task value and self-efficacy and their effect on achievement or motivation is accurately described by Green et al. (2017) who suggest that task value is central to the willingness to be engaged with knowledge while self-efficacy is central for preserving towards developing competence (Green et al., 2017). For instance, in respect with academic choice, studies converge that value beliefs are related with future enrolment choices while self-efficacy is related with students’ engagement once in the course (Neuville et al., 2007). However, Bandura (1986) asserts that students can drop out even though they have a high self-efficacy if they regard that the task or generally the learning process is of little value or when they believe that it has high costs.

Expectancy-value approaches have been recognised as valuable in realising the relationship between motivation and achievement. However, Goodenow (1993) points out that attention
should also be given to the social framework that expectancies, values and motivation take place and how these are moulded by the specific social situation. For instance, many researchers have found a relationship between students’ sense of belonging and motivation and particularly academic motivation (Goodenow, 1993), task goal orientation and intrinsic motivation (Anderman & Anderman, 1999).

The importance of belonging has been highlighted as early as 1943 by Maslow (1943) who introduced a hierarchy of needs in which he presents how humans’ motivation moves and includes the following terms in a sequence of importance: Physiological needs, Safety, Love/Belonging, Esteem, Self-actualisation. Maslow puts the sense of belonging as an essential need which precedes the esteem and self-actualisation. Goodenow (1993) argues that students, from all educational levels, have problems sustaining school and academic engagement in settings in which they do not have feelings of belongingness or school membership. In line with this, Juvonen (2006) emphasises that students’ perceptions about teachers’ support and attention are positively related with students’ motivation and engagement while these relationships have also been confirmed by other researchers as well (e.g. Wenzel, 1998). These research findings suggest that sense of belonging is identified as a critical factor of students’ retention, participation and engagement in school.

Indeed, the problem of students’ engagement in different school subjects is at the centre of educational researchers for many decades. As with the sense of belonging, another close construct that is often investigated, especially in science subjects, is the construct of identity and its relationship with students’ interests in school subjects.

In line with the sociocultural view, identity is fluid and as Faircloth (2009) suggests “identity is understood to be a negotiated, fluid, multifaceted sense of one’s perspectives and values,
a fluid and purposeful positioning, constantly shaped by the affordances and constraints of one’s context(s)” (Faircloth, 2009: 325-326). In other words, identity is understood as endorsed through an individual’s involvement in activities and discourses and constructed through perceptions of oneself and the participation of others as well (Carlone et al., 2014). This suggests that identities are also enhanced by the characteristics that other people in the same community attribute to us (Defelice, 2014).

For Burke and Stets (2009), identity expresses who an individual is regarding his/her role identities, group and social identities and personal identities. This organisation of identities into three layers assist us in realising how individuals connect to different facets of society, like the roles and groups’ participation, and how they are searching to find their individuality as humans (Stets & Serpe, 2009). Specifically, role identities refer to “a set of internalised meanings associated with a role” (Stets & Serpe, 2009:14). Group identities are defined as “meanings that emerge in interaction with a specific set of others (e.g. family, clubs)” (Stets & Serpe, 2009:15). It infers involvement with these groups and espousing the required behaviour for participation. Social identities are different in that they refer to “meanings associated with an individual’s identification with a social category” (Hogg, 2006). In other words, they refer to categories shaped by society for social stratification, e.g. ethnicity. Finally, person identities are “meanings that distinguish the person as a unique individual”, and not as an individual who holds a particular role or membership (Stets & Serpe, 2009:15).

While Erikson (1968) alleged that identity construction is a life-long procedure, he highlighted two critical periods in an individual’s life during which identity formation is commencing: adolescence and young adulthood. This specific age period has been the focus of science education researchers who attempt to understand students’ learning in school
settings and how identity is constructed. For example, Archer et al. (2010) posit that the devolution from elementary to middle school is a challenging transition period in which students’ science interest, motivation and engagement decline. Most of these studies, as Barton et al. (2013:37) argue, focus on the “how the process of becoming within a community of practice is reflective of one’s developing knowledge and practice within that community”. Therefore, science learning exceeds the cognitive side of science knowledge (Kim, 2018). Ideally, it involves learning about oneself and how to participate in science communities (Kim, 2018).

Of particular importance for researchers in this research area are also factors that influence students’ career choices. For instance, a comprehensive study of Stets et al. (2017) looked at factors that contribute to entering a science occupation for minority students. From these factors, their findings suggest that science identity was the only factor that influences students. Additionally, the factor that influences science identity was found to be students’ performance in their GPA: the higher their GPA performance, the more it raised students’ perceptions of being good scientists which eventually led them to choose science careers. This finding can be linked with what Wigfield (1994) pointed it out (it is already mentioned in the second paragraph of this subsection) about task value and that over time, children start to assign more value to tasks in which they perform well.

In the field of computing, few studies investigate computer science identity and how this can be constructed or enhanced. Among these studies, Wong (2017) argues that a computing identity for school children is hard to be constructed. He explains that most students are interested in doing computing, but few of them seek to be a computing person. One of the reasons for this is the typical stereotypes that see computer scientists as smart but yet lonely,
antisocial and male geeks. In the same vein, the study conducted by Master et al. (2016) sought to explore high school students gender differences regarding their interest in computer science and the role of stereotypes in this. Their results point out that girls’ lower interest in computer science may be contributed to stereotypes in this area. Their results further highlight that girls’ sense of belonging is impacted by stereotypes, which lead to a lower interest in the computing classroom.

Another study conducted by Mooney et al. (2018), explores factors that encourage female undergraduate students to enrol in computer science departments as well as the sense of belonging in the computing school. The results of their study suggest that female students demonstrate lower levels of sense of belonging than their male peers. Additionally, the factors that seem to influence women more for pursuing a BSc degree in computer science is job availability, salaries and job security. Among the least encouraging factors are “stereotypes of computer science students, university students’ advisors, career guidance not attached to the school, fitting in and career guidance teacher in school” (Mooney et al., 2018:91).

2.4 Concluding remarks

This chapter demonstrated a review of the literature regarding the three core theoretical frameworks underpin this research, namely, threshold concepts, misconceptions and affective dimensions of learning.

Regarding the threshold concept framework, the literature discussed studies in different subjects (including computer programming) that aimed to identify disciplinary threshold concepts by employing a variety of methods. However, despite thoughtful attempts to
identify threshold concepts within the disciplines, there are still concerns that focus on two primary directions: a rigorous methodological approach and the clarity of the threshold concept framework itself. Regarding the methodology, most concerns refer firstly, to the methods employed and the importance of highlighting the “qualitative differences” in participants’ understandings and secondly, to the participants themselves and who is regarded as most suitable to reveal these qualitative differences. Regarding the critique of the threshold concept framework, most concerns emphasise the ambiguity of the threshold characteristics while others question its existence.

The second section in the literature presented misconceptions and, mainly, it discussed misconceptions students experience in computer programming. The frameworks suggested so far, highlight different sources of misconceptions in programming, focusing both on students’ previous experiences but also on teachers’ practices. Interestingly, a connection between threshold concepts and misconceptions has not yet been established in the literature. However, inferences can be made regarding the troublesome aspect of threshold concepts and misconceptions, which is going to be examined further in the discussion section of this thesis.

The third part of the literature review focused on the role of affect in students’ learning. This section presented specific affective dimensions of learning, namely, self-efficacy, task value, motivation, sense of belonging and computer science identity. Specifically, the literature suggests that all these affective factors are critical constructs for students’ engagement in school tasks, for students’ performance and their career choices. Therefore, the question that arises and will be further investigated in this study is whether and how being in liminal space
affects students’ aforementioned affective dimensions of learning. This study aims to investigate this gap and offer some new directions in the threshold concept framework.
Chapter 3: Research Methodology

This chapter explains my research approach and describes the philosophical and methodological framework supporting the current research. Justifications are given for the design and methodological approaches employed to address the research questions. The chapter also provides information on the data collection, the data analysis steps carried out for each of the three phases of the research and information about my role as a researcher in each phase. The chapter concludes by examining validity issues and the ethical considerations of the research.

3.1 Philosophical Assumptions

Before starting my investigation, it was necessary for me to examine the philosophical assumptions underlying the practical level of my research, that is, the defined principles and values that I, as a researcher, should consider for my actions and beliefs. Thomas Kuhn (1970) refers to these principles as paradigms – the core values and the logical way in which research is based (Kuhn, 1970) – while Guba (1990) describes paradigm as a framework based on beliefs and principles about how the world should be considered (Guba, 1990).

Reflecting on a variety of readings and oscillating between different paradigms that could shape my research, I admit it was a challenging process. Hammersly (1996) views that selecting between the paradigms necessitates judgment of the circumstances instead of a commitment to a specific philosophical position. In view of this, I decided that it is best to focus on the research problem and questions and decide, with the researcher’s role in mind, what approach would better explore and unravel the phenomenon in a specific context.
Comprehending the relationship between my view of reality and the meaning I attribute to knowledge was central in justifying the research design and methodology of this research.

As a researcher, I tend first to look at the research problem and questions, and then, with no specific paradigm influencing my judgement, I consider which data collection and analysis technique would better suit the research questions of the study. To achieve the purpose of this thesis, it was first important to explore how the phenomenon under study is experienced by the participants, that is the teachers and the students. Thus, my perspective of the problem under study was first to investigate threshold concepts by interpreting the participants’ constructed meaning of their experiences, what they see as significant or their feelings and views of what they see as troublesome knowledge in computer programming. However, as a researcher, I believe that when the participants are grouped in a specific way, some commonalities will occur that will be relevant to the phenomenon under study, and I will be able to reach a more realistic view of the phenomenon. Therefore, establishing relationships between the variables under examination was also important for me to reach a more profound understanding. Considering these, I recognised that the problem under study is complicated, and none of the qualitative and quantitative designs would alone uncover the phenomenon and all its complexity. To this end, I decided that a mixed-method design, particularly an exploratory sequential design, would better fit the research aims. Sequential designs employ first qualitative data analysis methods to inform the quantitative phase which aims at finding more evidence to support the first findings. Therefore, the quantitative data are formed by the qualitative analysis conducted at the beginning of the research (Cresswell, 2014).

This specific way of designing my research is in line with the pragmatic paradigm. Specifically, pragmatist researchers focus on the research problem and research questions to
determine their research design. If the research problem is as complex as this thesis and does not imply a specific type of paradigm that is suitable, pragmatist researchers will argue for the integration of different methods and types of knowledge to understand and investigate the research problem (Saunders et al., 2009).

Pragmatism asserts that there are singular and multiple realities, avoids the debate about truth and focuses on explaining problems of the real world (Creswell & Plano Clark, 2007; Dewey, 1925, cited in Feilzer, 2010). Pragmatism stems from the work of many philosophers such as Peirce, James and Dewey (Creswell, 2014). It is focused on the research problem and regarded as the philosophical underpinning for mixed-method research in which the use of pluralistic approaches to generate knowledge is employed (Creswell, 2014). Pragmatists adopt multiple methods and different worldviews and assumptions (Creswell, 2014).

As a pragmatist, therefore, I recognise that there are multiple ways to explore and interpret the phenomenon under investigation and the world and that there is no particular and unique perspective that can reveal the whole picture and complexity of a problem. It should be noted here that being a pragmatist does not suggest that mixed approaches to research will always be employed; rather, pragmatists adopt the approach that they believe is most suitable for the research problem and can generate reliable results (Saunders et al., 2009). As Gilbert (2010) argues, the fundamental focus of pragmatism is the research’s outcome and not the focus on a particular worldview.

The pragmatism paradigm is flexible enough to allow the researcher to adopt elements of other paradigms, as the purpose is to employ methods that would best fit the research. According to Morrow (2007), paradigms can intersect if the research questions and data point to that direction. In other words, a research design based on pragmatism can
acknowledge the values of interpretivism and positivism. Therefore, the pragmatic paradigm supposes a combination of epistemologies to be employed (Gilbert, 2010) and, thus, this specific study adopts elements of constructivism and post-positivism.

Specifically, for the first and second research phase of the study, a constructivist approach leads the exploration. This is because I am examining threshold concepts in computer programming, and I am trying to understand them through the individual meaning that each participant (teachers) allocates to them. Constructivism differentiates from other theories of cognition in that knowledge is not a process of representing a real-world, an independent reality, but it plays an adaptive role (Glasersfeld, 2013). Constructivism rejects the objectivist view that reality exists independently of the researcher. For constructivism, knowledge is described as growing, progressive, developmental, subjective and workable constructed accounts by individuals involved in the generation of meaning in social environments (Fosnot, 2013). In other words, individuals construct their meaning, which is generated by their interactions with the world; meaning is constructed and not revealed, and, thus, multiple accounts of a phenomenon can exist (Gray, 1997). It is this individual meaning and the multiple understandings that teachers attribute to threshold concepts that are the focus of this study.

However, constructivism congregates under its term a variety of different views (Duffy & Cunningham, 1996). Cobb (1994) depicts this diversity by referencing to individual cognitive constructivism and sociocultural. The former, which stems from Piaget’s theory, highlights the “constructive activity” of the individual who attempts to understand the world while the latter connects activity to the individual’s involvement in “culturally organised practices” (Cobb, 1994:14). Cobb further argues that the sociocultural perspective puts at
the centre of learning the social dimension of consciousness and as such sees as secondary the individual one. As a result, from an educational perspective, the focus is on practices and approaches that will enable learners to engage in social activities. On the contrary, cognitive constructivism focuses on “the individual’s interpretive activity”, an individual’s cognitive self-organisation, and views classroom exchanges as a developing “microculture” that is dependent on the teacher’s and student’s efforts to organise their activities (Cobb, 1994:15).

I will not try to position myself in one of the approaches above because at least in the current study, I can see these two approaches as complementary. On the one hand, computer programming is seen by many researchers in the field as a source of cognitive skills, and thus, many studies from the early ’80s focused on exploring the cognitive skills that students need for practicing programming and other skills that are enhanced and further developed by programming. As such, the very nature of this subject focuses on the students’ cognitive processes constructed during practice. On the other hand, learning programming is situated in a classroom environment and surrounded by socio-cultural practices in which students can act on and learn from each other. Therefore, students can learn and transform their understanding through their interaction with peers. Considering these, I would argue that the first two phases of the study take a constructivist approach based both on the individual’s constructed reality and self-organisation but also acknowledging the influence of the social environment.

In this view, employing a constructivist approach in this phase of the research gave the participants the opportunity to discuss their experiences and thoughts on the difficult concepts of computer programming, what constitutes troublesome knowledge and how they experience such knowledge. The central point of interest in this study is the understanding
of the different experiences and actions of the participants (Fossey et al., 2002) through the
meanings they assign to them. Of course, there are limitations to this approach, such as the
lack of generalisation of the results in different environments and the researchers’
subjectivity, which may lead to preconceived results (Kim, 2003). Additionally, as
Denscombe (2002) states, the weakness of interpretive approaches is the lack of confidence
in results, as it is difficult to achieve validity and reliability (Denscombe, 2002). However,
a generalisation of the research results is not the purpose of this phase; rather, it explores
complicated phenomena (teaching threshold concepts) from humans’ life experience as they
interact with their social environment (classroom).

In contrast, the third phase of the research problem calls for a more post-positivist approach.
The main argument is that this part, based on the findings of the previous phases, is trying
to present commonalities between the participants that could potentially be generalised and
examines the relationship between variables. In this approach, knowledge stems from
employing quantifiable measures to examine the phenomenon (Cousin, 2002). In particular,
I am trying to understand if and how being in liminal space can affect students’ levels of
self-efficacy in programming, the task value they allocate in computer programming, their
motivation, computer science identity and sense of belonging in the computing classroom.
In this endeavour, I establish research hypotheses, which I test by conducting appropriate
statistical methods focusing on discovering relationships of cost and effect. Although I deem
this approach suitable for my research problem, I acknowledge that there are limitations I
must consider before establishing my conclusions. The most important limitation is that this
approach does not consider the social environment in which humans act – in this case,
students – and thus, environmental and other social and compound factors are disregarded.
Having established the ontological and epistemological views of the current thesis, I will continue by describing the specific methodological choices for each phase of the research.

3.2 Methodological Approach

As I have already discussed in Chapter 1, the study is separated into three distinct phases: the first one, aims to identify the concepts that computing teachers agree on being thresholds, the second phase aims to explore teachers’ experiences teaching these concepts and the third aims to explore the potential impact of liminality on students’ affective domain as well as identify students’ misconceptions in functions. For each phase, a different method is employed that suitably addresses the research questions of the corresponding phase. Thus, this section is separated into three subsections explaining each methodology employed and why this was considered suitable.

3.2.1 Phase 1a: Identifying threshold concepts – The Delphi approach

To address the first research question, I considered it best to employ a consensus technique such as the Delphi or nominal group technique. As I have discussed in the literature chapter, some researchers agree that in the search for threshold concepts, a consensus technique is needed (e.g. Baradell, 2013) while Shinners-Kennedy and Fincher (2013) advocate an exploration of teachers’ practice. Considering these issues, I decided that a consensus technique, and particularly the Delphi method, in which computing teachers form the panel of experts, would be advantageous for identifying potential threshold concepts. In the following paragraphs, I provide a brief description of the Delphi technique and how this was employed in the current study. A more detailed description is provided in the Data Collection and Analysis section of this chapter.
3.2.1.1 The Delphi method

The Delphi design belongs to the group of consensus development techniques. It is usually adopted when the existing evidence on a topic of interest is limited or where the collective opinion of experts would be beneficial (Hejblum et al., 2008) especially in research where the topic under investigation is full of uncertainty (Yang et al., 2012). Taking these factors into account, I considered that the Delphi method was appropriate for the first phase of this research: Firstly, functions is a subject area that has not been investigated much in computing education, and thus, existing evidence on this topic is limited. Although there is research indicating that this is an area that accumulates many of the students’ misconceptions in programming (Fleury, 1991; Madison & Gifford, 1997), potential threshold concepts in this area have not yet been identified. Secondly, accumulating the collective opinions of experts, in this case experienced computing teachers, has been identified as a gap in the literature of threshold concepts and computer programming and as a necessity for some researchers (Shinnners-Kennedy & Fincher, 2013).

The basic design of the Delphi technique includes bringing together groups of experts, not restricted in a geographical area, who will participate in several rounds answering a specific question through e-mail (Linstone & Turoff, 2002). After each round, the participants obtain feedback on the groups’ responses to the previous round (Avella, 2016). The rounds procedure is repeated until consensus is reached (Linstone & Turiff, 2002). I should note here that consensus does not mean a 100% unanimous agreement; according to Vernon (2009), consensus usually ranges from 55% to 100% with the most frequent occurrence being 70%.

The Delphi technique is mainly qualitative in nature, but it can also provide quantitative evidence depending on how it will be applied (Bourgeois et al., 2009). In the current study,
both a qualitative and a quantitative Delphi design were used. Specifically, a qualitative analysis was performed between the first and second round to organise the participants’ suggestions of threshold concepts. Quantitative analysis was used to calculate the consensus and stability of the Delphi process. To this end, a variety of measures and statistical analysis techniques were used and are explained in more detail in the Data Collection and Analysis subsection of this chapter.

For this phase, I employed the conventional Delphi design. A researcher can choose from two main Delphi designs: the conventional Delphi and the modified Delphi. The main difference is that in the conventional Delphi, the list of arguments, suggestions and generally the list of alternatives are the product of the panel’s initiatives while in the modified Delphi, the list is provided by the researcher (Avella, 2016). Thus, this study adopts the conventional Delphi design for two reasons: firstly, I was interested in exploring the participants’ suggestions of the concepts that from their experience are potential threshold concepts, and secondly, there are currently no studies that suggest potential threshold concepts in the thematic area of functions that could have been used to generate the alternatives.

3.2.1.2 Characteristics of the Delphi study

When using the Delphi design, I was careful to consider the important characteristics of this approach: anonymity, iteration and feedback. Anonymity is one of the main characteristics of this method and one of the main reasons that I chose to use this technique. I was interested in collecting the teachers’ individual suggestions on potential threshold concepts without considering one participant’s potential influence on another. Many studies advocate that opinions collected separately are more accurate than those from groups of people working together, such as focus groups (Dalkey & Helmer, 1963). This was one of the main reasons
that I did not employ focus groups or the nominal group technique. In effect, Dalkey and Helmer (1963:459) argue that this method seems “to be more conducive to independent thought”. This is because working individually gives participants the opportunity to express themselves freely without the influence of other people’s opinions (Bourgeois et al., 2009; Hsu & Sandford, 2007) and also ensures confidentiality between the participant and the researcher. Thus, anonymity was one of the first things I highlighted when I first contacted the participants. I also explained that this is a process where their arguments will not be associated with them, and no one except me will have access to their individual responses. Anonymity was achieved by contacting each participant individually and by ensuring that the debate and arguments presented in each round are independent of the participants’ reputation.

The second characteristic is the iteration process. In this study, a three-round Delphi was conducted until consensus and stability were reached (a more detailed description of the process is given in the Data Collection and Analysis section). Hsu and Sandford (2007) highlight the important process of iteration and argue that with several iterations, the group focuses more on problem-solving and on providing more astute opinions. Moreover, the iteration process minimises the effects of noise, which refers to the process of communication taking place in a group and which alters the information while dealing with individual interests not related to the research aim (Dalkey, 1972).

Finally, the third characteristic critical in this method is feedback. Indeed, for each round, I had to collect and analyse the data and return the feedback to the panel. The members were then asked to review the results and say if they agreed or not and explain why they propose changes. In Delphi studies, feedback is very important, as it provides the panel with the
opportunity to reassess initial standpoints and alter them if required (Hsu & Sandford, 2007). This characteristic was central in all three rounds of the study, and it was the process that required the most time. This is because, as ‘The Researcher’s Role and Challenges of Validity’ subsection explains, I had to be careful not to put my views and influence the process.

The list of potential threshold concepts from this phase formed the basis upon which the second phase of the research was conducted. The list was used to guide the discussion between me and the teachers and to explore their experiences while teaching these concepts. Section 3.2.2 describes in detail the methodology adopted for this purpose.

3.2.1.3 Participants

For the first phase of the study, the aim was to identify potential threshold concepts in functions. To achieve this, I considered that this investigation should be based on teachers’ practice and not on students’ suggestions as other studies did before. Along the same line with Shinners-Kennedy and Fincher (2013), I believe that the exploration of threshold concepts should focus on teachers’ practice rather than on students: it is difficult for students, especially those aged 14–16, to understand and express the kind of transformation they underwent once they understood a concept. On the contrary, teachers are witnessing their students’ transformation and are experts in identifying the troublesome aspects of the curriculum.

In Delphi studies, it is important that the researcher avoid selecting the discipline’s representative sample for the panel (Bourgeois et al., 2009; Avella, 2016). This is because selecting a representative sample is an aim in quantitative studies in which the results should
reflect the total population. In Delphi studies, however, the characteristic that must be secured is that of expertise and not a representative sample (Bourgeois et al., 2009). Actually, the aim is to involve members that can demonstrate deep levels of knowledge from their group (Avella, 2016).

In light of this, the first phase of the study is based on computing teachers. Two subgroups of computing teachers were considered for the study: the first one included those with more than five years of experience teaching at key stage 4 or 5, and the second group included those who had experience practicing programming at a professional level for more than seven years. I consider the latter subgroup as ‘expert teachers’ because they had both professional and teaching experience in programming.

Therefore, the basic characteristic that describes all the participants of this phase is that they are all computing teachers or previously employed as computing teachers in secondary education and have experience teaching at key stage 4 and/or 5. Another important characteristic is that 60% of the participants have practiced programming at a professional level for more than seven years. Including individuals with such expertise provided a variety of opinions, which is a necessary characteristic of the Delphi process. I should note here that initially, I had considered including expert practitioners with no teaching experience. However, considering Ludwig’s (1994) suggestion – that participants must have a personal interest to participate in a Delphi study – and that Delphi has a high dropout rate, I considered it best to include computing teachers with experience as practitioners.

Table 3 shows in detail the participants’ educational and professional background.
<table>
<thead>
<tr>
<th></th>
<th>Less than a year or none</th>
<th>1–3 years</th>
<th>4–6 years</th>
<th>More than 7 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>Years of teaching experience</td>
<td>0</td>
<td>10%</td>
<td>0</td>
<td>90%</td>
</tr>
<tr>
<td>Years of teaching experience at key stage 4 or 5</td>
<td>0</td>
<td>10%</td>
<td>10%</td>
<td>80%</td>
</tr>
<tr>
<td>Professional experience as programmer</td>
<td>10%</td>
<td>10%</td>
<td>20%</td>
<td>60%</td>
</tr>
</tbody>
</table>

To advertise the study, I created a call explaining the criteria for each subgroup and e-mailed it to UK master teachers and other computing teachers’ networks such as CAS London. In total, 13 individuals volunteered to participate. None of the participants had previously made known to the authors their thoughts and perspectives on the topic under research, and none of them had any relationship with the authors. This is an important prerequisite for Delphi studies. Specifically, an issue that the researcher should consider when selecting participants is potential bias, that is, the positions held by the participants may be acknowledged by the researcher. For this reason, Murphy et al. (1998) suggest that participants with a previous relationship with the researcher should be excluded.

### 3.2.1.4 Data analysis

The study was conducted from 14 March 2017 to 1 May 2017, and the main research tool to collect the data was questionnaires. This is usually the method that Delphi studies use. In the current study, the questionnaire included both open-ended and structured questions. In the following subsections, I describe each of the three rounds that were conducted while in the last section, I explain the statistical measures to determine consensus and stability.
Before the start of the Delphi rounds

Three days before the start of Round 1, I created a short literature review document on threshold concepts and encouraged the participants to read it and further investigate the topic by reading some papers on threshold concepts such as Meyer’s and Land’s articles. In the information sheet, I included information about the study and the timeline for each upcoming rounds of Delphi. I also asked the participants to complete the consent form and return it to me along with their Round 1 responses.

Delphi – Round 1

I e-mailed the first questionnaire separately to each participant and asked them to return it in a week. This questionnaire consisted of three parts: the first part included questions about the participants’ educational and professional background, the second part included questions about the participants’ perspectives on the threshold concept framework and on students’ difficulties in programming, and the third part included a task which asked participants to consider and propose threshold concepts in the area of functions, providing a short description for each concept. I sent the Round 1 questionnaire to 13 participants who had initially agreed to participate in the study, but only 10 of them sent it back.

After that, I started the analysis of the participants’ responses. Responses on parts one and two were quantitatively analysed with the SPSS software while responses on the third part were qualitatively analysed. The aim of the latter analysis was the generation of a list with all the potential threshold concepts proposed by the panel. Concepts with the same meaning but with a slightly different articulation were grouped together to form one potential threshold concept.
Delphi – Round 2

In the second questionnaire, I included all the potential threshold concepts suggested in the previous round along with the descriptions provided by the participants. The concepts were organised alphabetically. The questionnaire also included a task that asked participants to indicate their level of agreement of whether a concept is a threshold concept. For this purpose, I used a five-point Likert scale with items ranging from 'strongly disagree' to ‘strongly agree’. The second questionnaire was again e-mailed to each participant individually, and they were given one week to complete and return it. All participants returned the questionnaire, but they needed more time, and so this phase lasted almost three weeks.

The analysis of the panellists’ responses to the second questionnaire was a quantitative one. The purpose of this round was to investigate if consensus (whether a concept is regarded or not as a potential threshold concept) had been reached for some concepts (see subsection ‘Calculating Consensus and Stability’). As soon as the quantitative analysis was concluded, I created two lists: the first included the concepts for which consensus has been reached and the second one the concepts that have not. In each list, I included all the statistical information calculated to provide the participants with the corresponding feedback.

Delphi – Round 3

In the third questionnaire, I included the two lists above and the statistical information generated and asked the participants to review the information provided. In this stage, the participants were also asked if they wanted to change their previous responses based on the information provided for each concept. For this purpose, I included information about the
participants’ previous responses for each concept and asked them to add a new level of agreement if they wish to change their response and explain the reason for the change.

All the participants returned the questionnaires, and the analysis of the new responses was also quantitative, including the same statistical information as in the previous round. However, according to Dajani et al. (1979), to decide if the Delphi process can be terminated, stability must be reached for each concept. If stability is not achieved, then another round is needed. In this study, stability was achieved in this round, so the Delphi was terminated. As a final step, all the statistical information was disseminated to the participants.

**Determining consensus and stability**

In general, Delphi studies use both qualitative and quantitative measures for analysis depending on their design. Different researchers employ many different criteria to determine consensus. For example, Miller (2006) contends that agreement demands a specific percentage of votes in a certain range. As such, Ulschak (1983) sets the consensus criterion to 80% of the panellists’ votes being within two categories on a seven-point scale while Green (1982) sets the consensus criterion to 70% of the votes falling within three or higher on a four-point Likert scale while the median should be a minimum of 3.25.

To present information regarding the group’s consensus, most researchers use measures of central tendency and dispersion. The problem lies in which measure of central tendency is most suitable for each research study. On this view, some researchers argue that when Likert scales are used, the median is more adequate (Hill & Fowles, 1975; Jacobs, 1996). Additionally, Ludwig (1994) suggests that the mode is also useful; there are cases where the group’s decision is polarised, and thus, the use of mean or median may be deceptive.
For determining consensus, I followed Giannarou and Zervas’s (2014) suggestions, as I find that these are more reliable and accurate for the current study. Giannarou and Zervas (2014) note the insufficiency of the above measures when used alone to measure consensus. Alternatively, they propose the use of three measures that together would provide a better representation of consensus: (a) 51% of the participants’ responses in the ‘strongly agree’ category (between ‘agree’ and ‘strongly agree’), (b) the interquartile range (IQR) should be equal or less than 1 and (c) the standard deviation should be less than 1.5. The current study adopted these measures; the only deviation was that I increased the level of agreement percentage to 75% because the study’s sample was relatively small (10 panellists). By doing so, confidence in the results was increased.

Typically, researchers stop the Delphi process when a specific level of consensus is reached. Nonetheless, for some researchers, the consensus is regarded of low importance if group stability has not been achieved. Under this perspective, Dajani et al. (1979:84) consider that group stability is the criterion for stopping the Delphi process, and only after stability has been reached should the researcher proceed with measuring the level of agreement or consensus. They define stability as the “consistency of responses between successive rounds of a study” (p.84) which in statistical terms is translated as no statistical significance between two Delphi rounds (Von der Gracht, 2012). Similarly, Scheibe et al. (1975) agree that stability should be measured to determine when the Delphi process should be stopped.

There are two main directions to measure stability: individual stability, proposed by Chaffin and Talley (1980), and group stability, proposed by Dajani et al. (1979) and Scheibe et al. (1975). On the one hand, Chaffin and Talley (1980) argue that individual stability is more accurate in Delphi studies since group stability can be achieved despite important changes
in individual responses. In contrast, Dajani et al. (1979) contend that in Delphi studies, what is important is the group’s collective opinion and not the individual’s. In line with Dajani et al. (1979), in this study, group stability was used as a measure to indicate when the Delphi process should be stopped.

After deciding on group stability as the stopping criterion, another issue is what tests should be used to measure stability. One of the measures that Dajani et al. (1979) suggest is the chi-square ($\chi^2$), which is a nonparametric test used to examine the relationship between two variables. Contrary to Dajani et al. (1979), Yang (2003) argues that this test is used for two independent samples and is thus not valid in the case of Delphi where the group is the same. To this end, he proposes the McNemar chi-square test for dependent samples.

However, when researchers employ ordinal scales, the situation is changed. As such, Seagle and Iverson (2002) and De Vet et al. (2005) suggest using the Wilcoxon matched-pairs signed-ranks test, which is a nonparametric test comparing two dependent samples. Hence, it is regarded as suitable for Delphi studies, measuring before and after situations. For this reason, this is the test that I adopted, as ordinal scales were used to measure agreement in each proposed concept.

### 3.2.1.5 Researcher’s role and validity challenges

During the first phase of the research, my role focused on that of a coordinator and not a contributor as in the following phases. Avella (2016) contends that the Delphi researcher has a dual role: planner and facilitator. In the planning phase, I was responsible for identifying the members who will take part in the study, the number that will be sufficient and the specialty of each participant who will be invited to participate in the panel as well as
establishing the process of the communication. As a facilitator, my job concentrated more on taking control of the arguments presented in the panel. I ensured that the debate and arguments presented in each round were independent of the participants’ reputation and that all opinions and arguments receive equivalent weight.

As with all research methods, Delphi also has some defects that pose challenges on validity, and these mostly refer to the researcher rather than to the design. One of the first potential flaws is the researcher’s bias. Linstone and Turoff (2002) argue that the researcher may even accidentally influence the process because of their authority duties. For example, issues such as who participates in the panel group and how the research questions are framed are sources in which the researchers may put their own opinions and stances. This is why Avella (2016) argues that the best way to prevent this situation is to have an external reviewer to investigate the questions to ensure that the questions, as formulated, do not lead to specific responses. For this reason, my supervisor, as well as an external researcher, ensured that this was not the case in this research.

Another issue for me was my preconceptions. De Villiers et al. (2005) emphasise that the researchers’ preconceptions might influence how they interpret participants’ responses and how they transmit these to the participants for the next round (de Villiers et al., 2005). In the Delphi method, this issue is accelerated because the researcher transfers its interpretation as feedback to the participants. For this reason, I was very careful both in the way the feedback was created and in my individual communication with each participant. Thus, I was vigilant not to expose my perspectives or constrain the production of alternative positions but to trust the expertise of each member. Here, it was necessary for me to understand that my role is limited to that of a facilitator and not, in any case, that of a contributor (Avella, 2016).
Another challenge for me was the high dropout rates of Delphi studies. Hsu and Sandford (2007) argue that because of its iteration process, the Delphi method is vulnerable to low response rates. This is why it is important to accurately select participants who also have a personal motivation to take part in the study (Ludwig, 1994). However, even if I had selected participants with a personal interest, the response rate was decreased between the first round and the second (a dropout rate of 23%). In the other rounds, all the participants returned the questionnaires, but more time was needed to collect all the responses. Additionally, another problem with the iteration process is the actual time to complete the Delphi. The researcher may need a week or more to analyse, summarise the results and create the questionnaire for each round, which delays the whole process (Ludwig, 1994). This was another problem with the design of my study. The whole process lasted one month and a half while the actual time to complete the questionnaires was no more than 15 minutes each.

Finally, as a researcher, I had two other challenges with this design. The first was concerned with the anonymity factor. As I had argued, anonymity was one of the main reasons I adopted this design instead of the nominal group technique. However, as McKenna (1994) highlights, anonymity could tempt the participants to pay less attention and not to be fully motivated towards the purpose of the research. Sackman (1974) contends that anonymity is interpreted as a lack of accountability, and thus, participants’ opinions can be expressed without much consideration. This could potentially lead to less valid and rigorous contributions. Indeed, I believe that some of the participants did not dedicate the time that was needed to make the initial suggestions of threshold concepts. This is based on the actual descriptions of the concepts that they suggested, which were short or incomplete.
Therefore, for validity, I decided to further investigate the list with potential threshold concepts by adopting the IPA approach and interviewing the teachers. Based on Crotty’s (1994) and Dawson and Barker’s (1995) approach, which further included interviews with panellists to gain feedback on the Delphi results, I interviewed three panellists and one teacher not included in the Delphi process. As such, I was able to gain more feedback regarding the Delphi results and obtain deeper teachers’ experiences with these concepts. Additionally, as Powell (2003) suggests, I compared my findings with data from other sources (this is described in the Discussion section of this thesis).

3.2.2 Phase 2: Teachers’ perspectives and experiences with threshold concepts – An interpretative phenomenological analysis

To address the second-phase research questions, this part of the research focuses on the perspectives and experiences of teachers with the concepts identified as thresholds in the previous phase. Here, the emphasis was on describing and interpreting teachers’ lived experience with the potential threshold concepts and achieving a deeper understanding of the meaning of this experience. As such, by employing a phenomenological approach, teachers’ and students’ lived experiences can better be heard and described.

3.2.2.1 Interpretative phenomenological analysis

While the data analysis of this phase was planned as phenomenological, I felt that to reach a more profound understanding of threshold concepts experienced by the teachers, descriptions would not be enough. In view of these, and in seeking an analysis approach that goes beyond descriptions but keeps the phenomenological characteristics, I decided that the aim and purpose of this particular research phase fit the interpretive phenomenological
analysis (IPA), which, according to Charlick et al. (2016), is based on both descriptive and interpretive-hermeneutic phenomenology. IPA draws from three methodological areas – phenomenology, hermeneutics and idiography – that are combined to form the IPA study (Larkin & Thomson, 2011).

These methodological areas combined offered greater and new insights into the threshold concept framework itself and, specifically, how teachers and students interpret and experience these concepts in the classroom. To elaborate, understanding teachers’ and students’ experience with threshold concepts and examining how they make sense of these experiences subscribe to a phenomenological approach. Phenomenological research seeks to reveal and describe lived experiences and to achieve a deeper understanding of the meaning of experience (Moustakas, 1994; Husserl, 1970; van Manen, 1990). The key purpose is to identify reality from the participants’ descriptions of their experiences, emotions and feelings and to generate an in-depth and comprehensive description of the phenomenon (Moustakas, 1994). IPA is influenced both by descriptive and interpretative phenomenology (Charlick et al., 2016). For descriptive phenomenology, IPA uses bracketing, designates the “what and how” of the experience and generates descriptions of the participants’ experiences but does not provide explanations or any interpretations (Creswell, 2014, cited in Callary et al., 2015:63). Meanwhile, for the interpretive approach, IPA borrows Heidegger’s (1962) perspective: researchers or philosophers cannot step away from the world, and knowledge can only be reached through interpretation. Hence, in IPA, the interpretation of human experience and the meaning assigned to it are in relation to the researcher’s perceptions and views (Smith et al., 2009). In other words, IPA enriches the descriptions by adding an interpretative element to them.
Hermeneutics “is the study of understanding” (Collins and Selina, 1998:57). Influenced by hermeneutics, IPA always includes interpreting and making meaning out of participants’ lived experiences. This is because IPA views humans as being “sense-making creatures” (hermeneutics), and thus, their accounts will mirror their endeavours to make sense of their experience (Smith et al., 2009:3). It then lies in the researcher’s capacity to interpret the participants’ account and understand their experiences (Smith et al., 2009). In this respect, my interpretations were circumscribed on how the teachers were describing their experiences and, on my capacity to reflect and analyse. However, while trying to make sense of these experiences, I acknowledged that I carried my own bias. This, according to Smith (2008), may be an obstacle to the researcher’s interpretation and thus, Smith et al. (2009) suggest that “reflective practices and bracketing with a cyclical approach” is a necessity in IPA (Smith et al., 2009:35). Finlay (2005) refers to this as a dance between reduction and reflexivity. Thus, this necessitated a close view of what the participants were saying during the interviews while at the same time bracketing any of my presumptions. In the next phase, however, I acknowledged my preconceptions, and with an iterative process of reflection and going back to the participants’ exact words, I tried to base my interpretations on the participants’ world. Heidegger refers to this as a circular process in which the understanding of a text considers the individual parts together with the researcher’s interpretation of them and a further consideration of the full document (Sloan & Bowe, 2014).

Finally, my approach of investigation involves exploring how threshold concepts are experienced from individual to individual and not the generation of general truths or laws about how someone should exhibit these concepts or what kind of transformation someone must undergo through threshold concepts. Rather, my focus is on the individual case and
how different individuals experience this phenomenon. I assume that everyone is unique, and therefore, participants should be studied separately. Consequently, IPA, through its influence from idiography, offers a fertile approach for this: IPA concerns a comprehensive analysis of single cases and an investigation of the participants’ distinct perspectives (Pietkiewicz & Smith, 2014). This indicated an exploration of every single case first. Then, I moved on to the cautious investigation of similarities and differences across the cases and of meanings stemming from shared experience or concerns and then, if possible, to proceed to the development of general statements (Shinebourne, 2011:6).

Considering everything, I believe that IPA, as a method of analysis, was appropriate for exploring the teachers’ experiences with threshold concepts and the meaning they allocate to them. From my perspective, this required a thorough investigation of the way teachers make sense of their experiences with threshold concepts and their understandings and views (Reid, Flowers and Larkin, 2005). It also included me offering an interpretative account of the teachers’ and students’ meaning making of their experiences with threshold concepts during their practice. This allowed me to use experience common to the participants and me and summarise the experiences that are of interest in this study. In view of this, I recognised that I did not come to the interviews as neutral, but I already had an idea about what is that I want to know and what the participants may say and thus, as Høffding and Martiny (2016) point out, actively participate in the process of knowledge generation.
3.2.2.2 Participants

In this phase, the aim was to explore in more depth how teachers experience the concepts identified in the previous phase as potential thresholds. The underlying reason for this was to explore if indeed the concepts suggested by teachers in the previous phase have the characteristics of the threshold concepts. To achieve this, computing teachers again were considered suitable.

The decision to focus on teachers’ experiences as the primary emphasis of my investigation in this phase as well is based on an extensive literature review on threshold concepts and other researchers’ arguments and suggestions. While early research in this field focused on students’ experiences through phenomenological research, some researchers are sceptical of using students’ perspectives to address such a complicated phenomenon. Among them, Male and Ballie (2011:52) argue that threshold concepts can be identified through two sources: firstly, from students and secondly from “people whose experiences give them awareness of students’ experiences” such as teachers. They further support that teachers can report not only the troublesome aspects of threshold concepts but also the most important aspects, which are the transformative and integrative characteristic of these concepts – the focus of our current research. Shinners-Kennedy and Fincher (2013) align with this view and the use of teachers and suggest that threshold concepts can be identified from those who possess both pedagogical and content knowledge. They further support that research in this field has reached a dead end. They criticise the methods used for identifying threshold concepts in programming, emphasising that asking students about difficulties confronted in the past is an unreliable method. Finally, Zwaneveld et al. (2016) argue that this pedagogical and content knowledge cannot easily be found in undergraduate teachers; rather, they propose
secondary teachers mostly because they get more pedagogical training in comparison with undergraduate teachers and because their interaction with students is more frequent.

Aligned with the researchers above, I also consider that asking secondary students about the ontological and epistemological changes that they experience once specific concepts are understood may be an extremely difficult and unreliable endeavour, and therefore, my attention has focused on teachers whose experiences teaching these concepts can give us awareness of students’ experiences. It is the teacher’s job, through teaching, to observe and interpret and try to understand the difficulties that students encounter in the course as well as to understand when the students finally surpass their problems by observing changes in their attitudes, emotions, behaviours and performance. Allison et al. (1994:47) argue that “observing hold a key position in the cycle (observing, interpreting, decision making) because interpretation of classroom events and, consequently, pedagogical decisions are dependent on the observational abilities of the teacher”. In other words, teachers observe what is happening in their classrooms and how the students interact and communicate ideas, and then they interpret what they have observed and decide their next actions. It is these teachers’ interpretations of their daily classroom interactions and observations that I am trying to extract through their experiences teaching these concepts.

Phenomenological studies and IPA necessitate a quite homogenous group of individuals which denotes that the participants should demonstrate experience with the same phenomenon (Creswell, 2007). This is in contrast with grounded theorists who seek to include in their studies unusual cases to provide a compound theory based on multidimensional factors (Charmaz, 2006). In IPA, the researcher analyses the similarities or differences in a homogenous group, a group that is viewed as similar regarding some
characteristics (Pietkiewicz & Smith, 2014). For this reason, purposeful sampling or criterion-based selection is usually employed, given in this way the researcher with the responsibility to select participants with an important and meaningful experience of the phenomenon (Yüksel & Yıldırım, 2015).

Considering these issues, I chose purposive sampling to identify the participants. A purposive sample is a non-probability sample that is chosen based on the purpose of the study and on a particular population that can provide the information needed for the study (Teddlie & Yu, 2007). Thus, I selected the sample based on my judgement and the purpose of the research and looked for teachers with experience teaching computer programming at key stage 4 and/or 5.

Because IPA focuses on a thorough case exploration, the sample usually consists of a small number of individuals who will enable the detailed examination of each case. As generalisation is not the purpose of IPA studies, large samples are not advisable (Pietkiewicz & Smith, 2014) whereas, due to the idiographic characteristic of IPA, small sizes are suggested to prevent the loss of important meanings (Brocki & Wearden, 2006). As such, I determined the number of participants for this phase by considering Pietkiewicz and Smith’s (2014) criteria: “a) the depth of analysis of a single case, b) the richness of the individual cases c) how the researcher wants to compare and contrast single cases and d) the pragmatic restrictions one is working under” (Pietkiewicz and Smith, 2014:9). Considering these and Boyd’s (2001) point that a size of 2 to 10 participants are satisfactory to reach saturation, I initially aimed for four participants. From here, I checked for the depth and richness of the interview data and, because I found it to be adequate, no more participants were recruited.
As Pietkiewicz and Smith (2014) highlight, the researcher should focus on how deep the interview data is and not on the breadth of the sample.

All participants were computing teachers with more than seven years of teaching experience in programming at key stages 4 and 5. To keep the teachers’ participation anonymous, I have changed their names in this study: the teachers’ names were Rea, Olivia, Andrea, and Mateo. Specifically, Mateo and Olivia are master’s teachers and have worked as computing teachers for more than 10 years. They also have some years of experience working outside the school where they practiced programming for one to three years. Andrea and Rea have also taught computing in UK schools at key stage 4 and 5 for more than seven years. Both have experience practicing programming for more than four years outside of school settings. More details on the participants are provided in the Results section of this phase.

3.2.2.3 Data collection and analysis

Prior to the interviews

The most common data collection method of IPA is semi-structured interviews (Creswell, 2007). The aim of the interviews is to develop meaningful descriptions of the phenomenon as shared by the participants (Marshall & Rossman, 2006). The reason I chose semi-structured and not unstructured interviews was because with the former, I had the opportunity to prepare a set of questions beforehand. In addition, I acknowledged that other questions might emerge during the interview that may add more information about the lived experience of the participants. Except for interviews, other data collection methods I considered were focus group interviews (Yüksel and Yildirim, 2015). However, because my
research focuses on exploring the phenomenon from the individual’s perspective (idiographic), focus groups were not considered suitable for this purpose.

For the semi-structured interviews, I developed a ‘prompt sheet’ based on themes resulting from the first phase of this study. Thus, the questions referred to the concepts that the participants of the first phase have recognised as potential threshold concepts. These were the basis of my conversation with the interviewees. As I mentioned in the “participants” section, the participants were contacted via e-mail, where I explained the purpose of the study and that interviews could be conducted in their school, by phone, or by video chat. Two of the teachers selected the phone option, one teacher chose to visit me at my university, and another teacher chose the video option. All the participants signed the consent forms and sent them back to me through e-mail or in person before the interview.

The teachers’ interviews begun with me explaining my research and the result of the previous phase. I asked them to comment on these results, and a few moments were spent discussing these potential threshold concepts. After that, I started directing the conversation to their experiences as teachers with these potential threshold concepts. I also asked some follow-up questions that I had already prepared before the interview to elaborate on things that the participants were saying. The interviews lasted approximately about 35 to 70 minutes and continued until I felt that the participants could not share any more things on the topic.

Data analysis

After the data collection, I started the analysis by transcribing the participants’ records. Because all the audio recordings were transcribed by an external company (except one which was transcribed by me), I had to listen many times to the audio recordings and take notes on
the transcriptions. The analysis of IPA data can be complicated and time-consuming (Pietkiewicz & Smith, 2014). Also, it was difficult for me to be immersed in the data and provide the individual’s sense of the phenomenon and their meaning-making.

In short, IPA requires each verbatim transcript to be independently analysed. Consequently, I analysed each case separately, and themes were created with the hermeneutic cycle being an important part of the analysis. The analysis had two specific aims: first, to recognise and describe the participants’ world (Larkin et al., 2006), and second, to provide interpretations on how the participants make sense of this world. Having analysed all the cases individually, the last step was to compare the cases in search of similarities and differences and with the purpose to provide an overall description of the phenomenon. The following paragraphs detail the steps followed in this phase.

Multiple reading and making notes

I began the analysis by listening to the audio recordings and reading the transcribed interviews several times. As Smith et al. (1999) point out, IPA analysis focuses on reading and re-reading the transcripts. This helped me to be submerged in the data, to remember the interview atmosphere and to take notes on the transcript. Specifically, these notes can centre around content, language use, context, emotions and the researcher’s reflection and interpretation (Shinebourne, 2011).

Transforming notes into emergent themes

To identify themes, I worked with my notes rather than the transcripts. Here, the purpose was to convert my initial comments into emergent themes which represent an abstract level but are still grounded in the individual’s case. In doing this, I also sometimes went back and
re-read the transcript, which, according to Shinebourne (2011), is a demonstration of the hermeneutic circle.

Seeking relationships and clustering themes

The next step was to search for connections between the themes and try to group them into clusters with each one containing themes with some “conceptual similarities” (Shinebourne, 2011:18). Some of the themes may be abandoned if they do not fit the clusters. Thus, the outcome of this phase was to arrive at super-themes with subthemes and links to the lines of the transcript and a short description of the subthemes.

Writing up

Before the start of the writing process, I described each theme and illustrated extracts from the transcripts to help the readers evaluate the relevance of the interpretations and to present the participant’s voice. Thus, this phase includes not only the participant’s case in their own words but also my interpretations. This process is repeated for each case, and, thus, I was vigilant to view each case separately and bracket the outcomes of the previous ones. As soon as I had analysed all the transcripts, I constructed a final table – a consolidated list – including themes for the study as a whole. This again was an iterative process which required me going back and forth in the transcripts and merging themes or rejecting themes depending on their capacity to provide information that would enhance the account as a whole (Shinebourne, 2011). This table provided all the information that I needed to generate the narrative of the study, which includes an interchange between the individuals’ account – the participants’ own words – and my interpretative stance. By doing that, the narratives include the participants’ voice while enabling the reader to evaluate the accuracy of my interpretations.
3.2.2.4 Researcher’s role and validity challenges

The second phase of the study included a change in my role as a researcher. In this phase, my role was not confined to that of a facilitator as in the Delphi study but included a more interpretative stance on my behalf.

Thus, in this phase, I was involved in an interpretative process which focused on the participants’ meaning making of their experiences. This is the reason why IPA includes a double hermeneutic: the researcher tries to understand the participants’ meaning of their experiences (Smith, 2004). Smith (2004) emphasises that the researcher must interpret data through their perspective when creating themes. Thus, in the writing process, my experiences were part of my interpretation of the participants’ meaning making of their experience. However, as Larkin et al. (2008) suggest, my interpretations should be grounded on the participants’ reports. I was careful to ground my interpretations on the teachers’ account. On the contrary, during the data collection phase, I was involved with the bracketing process of phenomenology to obtain an accurate representation of the teachers’ descriptions and meanings of the phenomenon. The bracketing process was also useful to me when I moved from case to case because it helped me come in each case with a clear mind and investigate it separately from the other ones. The reflective journal I kept helped me clear up my thoughts by writing down my views on each case and acknowledging my preconceptions before moving on to the next case. For the analysis of each case, I also found it helpful to wait 3–4 days before moving on to the next case.

The challenges that I encountered with this approach centred around the researcher’s bias as with most of the qualitative studies. In the generation of themes and then my interpretations, following Collins and Nicolson’s (2002) suggestions, I tried to minimise my bias by reading
the transcripts many times to make sure that my interpretations are indeed based on the participants’ account. The researcher must ensure that each theme is actually based on the transcripts (Brocki & Wearden, 2006) while Smith and Osborn (2003) advocate for a distinction between the participants’ accounts and the interpretations of the researcher. For this reason, as Brocki and Wearden (2006) suggest, I based my interpretation on the criterion of “grounding in examples”, giving the opportunity to readers to evaluate my interpretations. Following Flowers et al.’s (2000) recommendation, I tried to provide the most representative extracts for each theme produced.

Other challenges stem again from the qualitative nature of this approach, which raises issues of validity and reliability (Golsworthy & Coyle, 2001). Some researchers validate their interpretations by using other researchers and academics to check their interpretations (Brocki and Wearden, 2006). Additionally, the transcripts can also be analysed by several researchers autonomously and then joined themes would be agreed on, while others would ask the participants to validate the interpretations (Brocki & Wearden, 2006). Finally, Yardley (2000) points out that reliability is not a criterion for qualitative studies which seek to provide some interpretations of many possible ones while Osborn and Smith (1998) argue that the aim of validity in qualitative studies is “to ensure the credibility of the final account” (Brocki & Wearden, 2006:31).
3.2.3 Phase 3: Students’ affective dimensions in liminal and post-liminal space and students’ misconceptions in functions

3.2.3.1 Phase 3a: A correlation and causal-comparative design and regression analysis

In the third phase of my research, the main aim was to investigate if students who demonstrate evidence of being in liminal space experience different levels of the affective constructs under investigation than students who show evidence of being in post-liminal space. To this end, this phase of the study employs a correlation and causal-comparative research design.

A causal-comparative design is employed when the researcher is interested in finding relationships between independent and dependent variables. Specifically, researchers who use this design compare two or more groups (usually split by the independent variable) with the aim to identify whether the independent variable affected the outcome. Gay et al. (2012:227) state that “the basic causal-comparative design involves selecting two groups that differ on some variable of interest and comparing them on some dependent variable”. However, Lochmiller and Lester (2015) highlight that researchers who use this design should not attempt to establish a cause-and-effect relationship.

Correlation designs are used to investigate how two or more variables are related; specifically, correlation designs look at how the values of the examined variables differ from each other (Mayers, 2013). Therefore, although both causal-comparative and correlation designs endeavour to determine relationships between variables, neither approach can firmly state cause-and-effect relationships.
In this phase of my research, the causal-comparative design was used to compare the levels of self-efficacy, task value, calibration, sense of belonging, computing identity and motivation of two groups: students with evidence of being in liminal space and students with evidence of being in a post-liminal space. In this endeavour, I also considered the relationships that the variables above form with each other in each group, and thus, a correlation design was also employed.

Additionally, the study would also explore the nature of the relationship between the liminal state (dependent variable) and one or more independent variables (self-evaluation, self-efficacy, task value, sense of belonging, identity and motivation) by employing logistic regression. Logistic regression is specifically employed when the researcher is interested in predicting the relationship between independent variables and a binary dependent variable (e.g., liminal or post-liminal). By investigating this relationship, it may be possible to suggest an alternative way (quantitative in nature) for identifying if a student is probably in a liminal state in a particular part of the curriculum. In other words, it may be possible to predict if students are in liminality by considering students’ affect. Exploring a model of affective dimensions that can predict (at least to some degree) students’ liminal state, could be of great use for teachers who would like to monitor their students’ progress and understand why their students are stuck in a specific part of the curriculum.

To create the two groups – students in liminal and in post-liminal space – I took into consideration the suggestions of Hamm (2016). In his research, he employed the SOLO taxonomy to identify students in liminal space. Based on Angelopoulou and Vidalis (2014) and Lucas and Mladenovic (2006), Hamm (2016) highlights that qualitative differences in students’ understandings are evident between two levels of the SOLO taxonomy: multi-
structural and relational stages. Specifically, Angelopoulou and Vidalis (2014) emphasise that students’ appreciation of the subject area also begins from the relational level while Ashwin (2008) argues that students who cannot progress beyond the multi-structural level demonstrate surface knowledge (Angelopoulou & Vidalis, 2014; Ashwin, 2008, cited in Hamm, 2016). Therefore, in this research, students who are still in the multi-structural level in functions will be allocated to the liminal group whereas those who demonstrate evidence of being in the relational level – a level where “a transformative quality of acquiring a threshold concept is evident” (Hamm, 2016:56) – will be allocated to the post-liminal group.

Figure 3“A visual representation of the acquisition of a threshold concept with respect to the levels of learning outcomes in the SOLO taxonomy: (a) when students cannot cross the threshold and (b) when students cross the threshold”. (Hamm, 2016:150)

I should note here that for a student, not reaching the relational level, in general, does not indicate that the student is in liminal space, that is, not all students that are limited in a surface level of understanding experience liminality. Therefore, in this research, I specifically
followed two pre-conditions that must be in place before making any conclusions for a student: first, the student must be exposed to the subject area for enough time in order for them to experience the liminal space as described in the literature chapter, and second, the area under examination should have been identified as a threshold area or an area with threshold concepts. Both these conditions were met in my research design, as all the students had experience with functions in programming, and most significantly, based on the results of the previous phase, the area of functions has been suggested by the computing teachers as an area with threshold concepts (see results of phase 2).

Considering everything, being in liminal space or not was treated as the independent variable whereas self-efficacy, task value, calibration, sense of belonging, computing identity and motivation were treated as the dependent variables.

3.2.3.2 Phase 3b: Students’ misconceptions in functions: Content analysis

In the final part of the third phase of the research, the aim was to explore students’ misconceptions in functions and to investigate if there is a link between threshold concepts and students’ misconceptions. To this end, qualitative analysis, and specifically, content analysis, was employed.

Content analysis is a systematic coding and categorising approach to investigate trends and patterns, their frequency and relationships in textual information (Vaismoradi et al., 2013). Incorporating content analysis in computer programming is not a direct process. Shah et al. (2017) presented a qualitative content analysis approach for programming errors, which includes four steps: paraphrasing, generalisation, categorisation and check. Paraphrasing is a process that removes the code components not related to the error (not content-bearing).
Generalisation involves the description of the error in natural language, and categorisation includes the process of producing more abstract versions of the generalisations made previously to produce categories that include similar errors. Finally, the check refers to the validation process of the errors and the category to which they have been assigned.

The process above was followed in this thesis to identify students’ misconceptions, and this is described in detail in the Data Collection and Analysis section.

### 3.2.3.3 Participants

For the two final phases of my research, students at key stage 5 were considered. Because in causal-comparative designs the independent variable is used to separate the groups, internal validity is often threatened. Therefore, to strengthen the design, I only considered homogenous samples.

To advertise the study, I created a call, explaining the criteria to students who should take part in the research, and e-mailed it to UK master teachers and other computing teachers’ networks such as CAS London. Therefore, I did not advertise the call to students directly but rather to their teachers. The basic criteria I listed were that the students should have experience with functions in any programming language, and they should have selected programming for their AS/A levels. In total, 123 students took part in the study from seven different UK schools. The following table shows the number of students per school that participated in the research.
Table 3.1 Number of students per school

<table>
<thead>
<tr>
<th>School</th>
<th>Number of students</th>
<th>Boys</th>
<th>Girls</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>51</td>
<td>37</td>
<td>14</td>
</tr>
<tr>
<td>S2</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>19</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>S4</td>
<td>15</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>S5</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>S6</td>
<td>15</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>S7</td>
<td>8</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>total</td>
<td>123</td>
<td>98</td>
<td>25</td>
</tr>
</tbody>
</table>

Almost all the students had experience in Python, and therefore, they participated in tasks written in Python. Only in one school (number 3 in the table) students had experience in C#, and therefore, all tasks were written in C# for this group.

3.2.3.4 Data collection and analysis

Phase 3 focuses on students and whether being in a liminal space can affect students’ affective dimensions of learning. With a correlation and causal-comparative design, the most suitable data collection technique is structured questionnaires and tests. These two were the fundamental data collection tools used in this phase of the study.

Questionnaires as research instruments are used to collect information on the participants’ attitudes, beliefs and other cognitive or non-cognitive constructs (Conway, 2006). While questionnaires can be used both in quantitative and qualitative studies, the latter usually uses open-ended forms of this research tool, whereas the former uses close-ended questionnaires. Therefore, in this study, close-ended questionnaires were employed to gather information
about the participants’ levels of self-efficacy, task value, sense of belonging, motivation and computing identity.

Specifically, the questionnaire used for the first two constructs – self-efficacy and task value – was the Motivated Strategies for Learning Questionnaire developed by Pintrich et al. (1991). The questionnaire aims to measure students’ motivation and self-regulation strategies and is comprised of 81 items grouped in sections, each covering a different dimension of motivational constructs and self-regulation learning aspects. For the aims of this study, I employed the subsections referring to students’ self-efficacy and task value. Particularly, the self-efficacy section consists of eight questions with a seven-point Likert-scale (a=.93) while the task value consists of six questions (a=.90). Cronbach’s value in both questionnaires indicates a very good reliability. Scores on both these sub-questionnaires require a calculation of the mean of the items.

The second questionnaire that was used in the research was the BASICS questionnaire, developed in 2016 as part of the computer science 100005 program funded by the National Science Foundation. One of the aims of the BASICS study was to create reliable tools for the computer science community. The questionnaire is comprised of three major sections. The parts that were used for this research were computer science identity questions (three questions in total) with a Cronbach’s a=.84, motivation for computing class with a=.96 (four questions) and sense of belonging questions with a=.95 (four questions). All the questions

---

5 “The Barriers and Supports to Implementing Computer Science (BASICS) study is a three-year exploratory research project funded by the National Science Foundation (#1339256) as part of the CS10K program—an ambitious effort to have 10,000 well-trained computer science teachers in 10,000 schools” (p. 1). More information at https://s3.amazonaws.com/cemse/basics/files/findings/BASICS_SQ_Descriptives_FINAL.pdf
were based on a six-point Likert scale. In the end, I also asked students to evaluate themselves on an 11-point Likert scale to calculate their calibration ability.

To identify whether students demonstrate evidence of liminality and their misconceptions in functions, I decided to create programming tasks that focus explicitly on functions and the difficulties that the computer teachers in the previous phase highlighted. Therefore, considering the results of the previous phase and what was already discussed in the methodological approach about the SOLO taxonomy and how it can be used to identify students in the liminal space, I created six programming tasks in total. These tasks focused on students’ knowledge in functions and included multiple-choice questions, find the errors, find the purpose/aim of the program, fill in the gaps, parson puzzles, and open-ended questions.

Specifically, the first exercise asked the students to match the concepts of parameters, arguments, return statement, function definition, calling a function and print statement with the appropriate annotated text in the given code. The aim here was to see if the students had a basic understanding of the concepts included in this part of the curriculum. The second exercise included a short program in Python or C#, which included a main and another function, and asked the students to identify and correct the errors (both syntax and logic). This exercise was a little more advanced than the first one, as it included logical errors in addition to syntactical ones, and thus, a deeper understanding of functions was required by the students. The third exercise asked the students to read a program written in Python or C# and answer two reading comprehension questions: ‘What is the purpose of the program?’ and ‘What will be the output of that program?’. The fourth exercise was a parson puzzle: the students were asked to put in the correct order a group of code statements in order for the
program to achieve its purpose (it was given in written form). The fifth exercise was a ‘fill in the blanks’ exercise focusing on parameters, arguments and return variables, and the sixth task asked the students to write from scratch a program that creates and calls a function that calculates the area of a triangle (all the tasks are presented in the Results section and in the Appendix of this thesis). The data were collected through an online survey tool, the link to which was given to students by their computing teachers. The test was completed during the computer science class with the supervision of their teachers.

When designing the assessment tool, I considered Scott’s (2003) argument that assessment tests should include questions from all six levels of Bloom’s taxonomy, as most of the students are able to respond to low-level questions and few can answer the most complex questions. This way, I had a more objective way to determine what each student had understood or not. The most influential study, however, was that of Shuhidan et al. (2009), which used Bloom’s taxonomy to classify the multiple-choice questions and SOLO taxonomy to evaluate students’ responses. In the same vein, in this assessment test, Bloom’s taxonomy was used to categorise the questions to Bloom’s different levels, and the SOLO taxonomy was used to evaluate students’ level of understanding and group the students to those in liminal and post-liminal spaces. Extra consideration was given to Whalley and Kasto’s (2013) paper, in which they categorise different types of programming tasks into Bloom’s and SOLO taxonomies. Building on these studies, I created the six programming tasks with increasing difficulty, which are presented in the Appendix.

Before administering the test and questionnaire to students, I created two versions. The first one had the questionnaire first and then the programming test, and the second one presented the programming test first and then the questionnaire. The reason for this differentiation was
that I was interested in exploring if and how students’ levels of affective dimensions of interest in this study change when they encounter troublesome knowledge.

The main purpose of this phase was to explore how students’ levels of the constructs under investigation may be affected by their liminality state. To explore this relationship, once the students’ responses were collected, I corrected their programming answers and defined the level of SOLO taxonomy that each student had reached. Those who reached the relational level were categorised to the post-liminal group, whereas the others were classified in the liminal group. For each student, I calculated their performance, self-efficacy score, task value score, sense of belonging, motivation, computing identity score and calibration bias and accuracy. All these data were used as inputs in the SPSS, where appropriate tests were run to answer the research questions.

In the second part of this phase, the emphasis was on exploring students’ common misunderstandings and misconceptions in functions. To this end, qualitative analysis and specifically, content analysis was used to investigate and identify misconceptions held by students. Gbrich (2007) argues that content analysis can be used both to qualitatively analyse the data and to produce quantitative information, which was important in this study. I was interested not only in producing the patterns of students’ errors but also on how often these errors occur and on how many students.

For this phase, I specifically followed the steps proposed by Shah et al. (2017). Once students’ responses on the tasks were collected, the first step was to locate students’ logical errors while removing – not considering – any syntactical errors. Along with this, I removed any code that was not related to students’ errors (paraphrasing phase). However, the paraphrasing phase did not remove many code components, as the tasks were short.
Therefore, this phase focused more on distinguishing between students’ misunderstandings and syntactical errors. The next phase was for me to produce the coder manual. In this process, I described students’ errors in natural language. Then, the next step was for me to create categories of students’ errors, that is, bringing together similar errors under one single category. Finally, to ensure the validity and reliability of the interpretation of the misconceptions, another researcher was involved in the process: my role was to produce the coder manual and create the categories of errors with students’ misconceptions in them while the second researcher used these categories to classify the students’ errors again. The inter-rater reliability of Cohen’s kappa was calculated, and a substantial agreement was found: $k=0.675$, $p < 0.001$ (approximate 95% confidence interval on kappa: 0.612 – 0.737).

### 3.2.3.5 Researcher’s role and validity challenges

In the third phase of the research, my role again turned into that of a coordinator rather than that of a contributor. In quantitative research, the participants act independently from the researcher, whose role is strictly to analyse the data objectively.

When conducting quantitative studies, one of the first concerns is the sample. Specifically, issues around the sample refer to the size and suitability of the participants. With regard to the suitability of the participants, this was not an issue in my study, as the students who participated met all the requirements I have listed in my call. However, the size of the sample may pose an issue in some cases. In correlation studies, Sekaran (2003) proposes that sizes between 30 to 500 are suitable for research, and thus, my sample of 123 students covers this criterion. However, Sekaran again points out that when the sample is grouped into subsamples, then the minimum size for its category needs to be 30. This criterion covers the subgroups of the liminal and post-liminal group.
Another issue stemming from the quantitative nature of the study was the validity and reliability of the instruments. To this end, I only searched for questionnaires that had good validity (Cronbach’s alpha) and have been used by other researchers as well. For data collection in the third and fourth phases, I had to be very careful in designing tasks and selecting questionnaires that can be used by other researchers, as quantitative studies must be replicable.

In quantitative studies, data analysis requires the researcher to choose the appropriate statistical measures and techniques to answer the research questions. Therefore, for each of the statistical measures and techniques in the third phase of the research, I made sure that the data fulfilled the criteria that these techniques impose (e.g. normality). To this end, in the Results section, I listed all the criteria for a corresponding technique and how the data conforms with or deviates from these. Finally, when interpreting the results, it was important for me to keep in mind that both correlation and causal-comparative studies do not firmly establish cause and effect. Rather, they determine relationships and describe whether the independent variable affected the outcome.

For the second part of the final phase, in which content analysis was employed, most of my concerns aligned with the ones described in the second phase of the research, as both phases are qualitative studies. Focusing on validity and reliability issues, Mayring (2003:45) points out that in qualitative analysis, content is more central than methodological issues, and thus, validity is more important than reliability. Nevertheless, to ensure the reliability of the coding, another researcher was also involved in this coding process, as discussed earlier in this chapter. Finally, for validity, as with the second phase of this research, I based my interpretation on the criterion of ‘grounding in examples’, providing readers with the
opportunity to evaluate my interpretations. To this end, the most representative examples for each of the students’ misconceptions are provided in the Results section (Flowers et al., 2000).

3.3 Ethical Considerations

Ethics approval is necessary for research that includes human participants for protecting human rights and well-being by minimising intellectual distress and the potential dangers stemming from the research and for protecting the researcher’s rights to perform any legitimate inquiry (Canterbury Christ Church, 2014). To this end, ethical principles have been established that can guide researchers in the design and conduct of research.

The code of ethics and conduct (British Psychological Society, 2018) is based on four ethical principles, each one defined in statements of values and important considerations: “respect, competence, responsibility and integrity” (p.4). The first one, respect, considers issues such as privacy and confidentiality, consent, shared values. Competence considers issues regarding the researchers’ skills. Responsibility considers issues regarding professional accountability and respect of the participants’ well-being. Finally, integrity focuses on issues such as honesty, unbiased representation, conflicts of interest and handling misconduct. Accordingly, the British Educational Research Association lists five generally recognised ethical principles, which involve “minimising harm, respecting autonomy, protecting privacy, offering reciprocity and treating people equitably” (Hammersley and Traianou, 2012:2-3).

Ethical approval for the study was given by the King’s College London Research Ethics Committee (LRS-17/18-4282). The current study was conducted by following the guidelines
from King’s College London Guidelines on Good Practice in Research and the guidelines from BERA (2011, Ethical Guidelines for Educational Research).

In all the phases of the study that involved teachers, I provided the consent forms along with information sheets to the participants prior to the interviews or the Delphi study. The information sheet detailed the aim and purpose of the study, in what processes the participants would be involved, and how their data will be stored, processed and by whom. I also highlighted that they have no obligation to take part in the study and that if they decide to participate, they have the right to withdraw at any time. For the Delphi study, the participants e-mailed back the consent forms signed while for the interviews, some participants e-mailed the consent forms, and others had given me the forms before the start of the interview.

For collecting the students’ data, I first informed the headteachers and the computing teachers about the study by sending them the information sheet describing the details above. As soon as the headteachers and the computing teachers agreed to take part in the study, everything was set for the study to start. When I created the online tests for the students, I attached the information sheet and the consent form on the first page of the test for the students. Permission for their parents was not needed since the students were over 16 years old. For confidentiality issues, the tests were completed anonymously.

Before the students started the tests, I informed them in my letter that if they feel any kind of discomfort or anxiety, they have the right to withdraw from the procedure. Researchers who include human subjects in their research should ensure that they will not cause harm, risk or any kind of anxiety and harassment to their subject that is greater than the ones they experience in their everyday life (Department of Children and Youth Affairs, 2012). Lucas
(2008) argues that to justify a research proposal which includes children, the risks must be at least equally scaled to the potential benefits of the child and society, and these risks should include any kind of discomfort as well as the psychological and growth aspects of the child.

All the participants were also informed about how their data will be processed. Specifically, I explained that their interview data will be transcribed and that quotes from their interviews will be used in my thesis and for publication purposes. Here, I also explained that all identifying evidence would be deleted, and, in any case, the data would not be linked to them. This information was also part of the information sheet that I distributed to them. For this reason, I used pseudonyms throughout my thesis.

3.4 Concluding remarks

The chapter presented here, described in detail the methodological approach of the research. The chapter first presented the philosophical assumptions and theoretical perspectives of this study and highlighted the pragmatic approach as the central paradigm leading this study. Particularly, it was emphasised that threshold concepts are a complex phenomenon, and it is important for researchers to employ a mixed methodological approach to identify, explore these concepts and interpret the participants’ experiences.

The chapter continues by presenting the methodology, and specifically the methods, the data collection and analysis techniques for each of the three phases of the research. Specifically, the first part of the research employed the Delphi method, which is a consensus technique. The aim was to investigate teachers’ perspectives on the threshold concept framework and create a consensus list with potential threshold concepts in functions. The second phase employed the interpretative phenomenological analysis method to investigate teachers’
experiences with teaching potential threshold concepts in functions. This section highlighted the major characteristics of this method and the reason why this approach was suitable for this study. It was argued that because of this framework’s complexity, it is necessary for the researcher to engage in an interpretation of the participants’ meaning making of their experiences with potential threshold concepts. The third phase of the study employed a causal comparative - correlational design (first part) and a qualitative design (content analysis-second part). The aim of the first part was to investigate how students experience liminality by taking into consideration affective dimensions of learning and by employing quantitative methods. The second part aimed to identify students’ misconceptions in functions and to explore therefore, potential links between misconceptions and threshold concepts.

Following this chapter, the results and the discussion of the results are presented for each phase of the thesis. Chapter 4 and 5 present the results and the discussion of the first phase of the research, chapter 6 and 7 present the results and the discussion of the second phase of the research and chapter 8, 9 and 10 present the results and the discussion of the third phase of the research.
4 Chapter 4: Results of Phase 1 - The Delphi study

This chapter presents the results of the first phase of the research\(^6\). The first phase of the study focuses on computer science teachers and investigates their perspectives on the threshold concept framework as well as their suggestions on potential threshold concepts in functions in computer programming. To this end, a consensus technique was employed and particularly the Delphi method and thus, the chapter presents the participants’ responses in each round of the Delphi study. In total, three rounds were conducted until stability and consensus were achieved.

4.1 Delphi - Round 1

The first questionnaire distributed to teachers included three sections: the participants’ education and professional information (the results were presented in section 3.4.1), the participants’ perceptions of the threshold concept framework the results of which are presented in the next section and suggestions of potential threshold concepts.

4.1.1 Perceptions of the threshold concept framework

The second section of the first questionnaire included questions about the panellists’ perspectives on the threshold concept framework and on students’ difficulties in programming. The results of each question are depicted in Table 4.1.

\(^6\) The results of the first phase of the research were published in the following paper which can be found in the Appendix: Kallia, M. and Sentance, S., (2017). Computing Teachers’ Perspectives on Threshold Concepts: Functions and Procedural Abstraction. In Proceedings of the 12th Workshop on Primary and Secondary Computing Education (pp. 15-24). ACM
<table>
<thead>
<tr>
<th>Question</th>
<th>Yes</th>
<th>No</th>
<th>I am not sure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Are you familiar with the framework of threshold concepts?</td>
<td>90%</td>
<td>10%</td>
<td>0</td>
</tr>
<tr>
<td>2. Do you believe that the framework of threshold concepts can explain some of the students’ difficulties in computer programming?</td>
<td>80%</td>
<td>0</td>
<td>20%</td>
</tr>
<tr>
<td>3. Would you change the way you approach a concept or a construct if you knew that it was a threshold concept?</td>
<td>70%</td>
<td>10%</td>
<td>20</td>
</tr>
<tr>
<td>4. Do you think more research is needed in how core concepts or constructs should be approached in teaching computer programming at KS4 and KS5?</td>
<td>100%</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5. Do you think that some students can effectively write code even though they haven’t really grasped the theory behind the concepts they employ?</td>
<td>40%</td>
<td>50%</td>
<td>10%</td>
</tr>
<tr>
<td>6. Do you think that some students have difficulties in applying their understanding in programming tasks even though they have a theoretical understanding of the corresponding concepts and constructs?</td>
<td>100%</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7. Do you think that students encounter more difficulties in applying their knowledge in programming tasks or in understanding concepts and how constructs work?</td>
<td>Applying 30%</td>
<td>Understanding 20%</td>
<td>Both 50%</td>
</tr>
</tbody>
</table>
Questions one to three refer directly to the framework of threshold concepts, and it seems that most of the participants were familiar with this framework. Most of them also indicated that threshold concepts can explain students’ difficulties in programming and that they would change the way they approach and teach a concept if they knew that it is a threshold one. Questions five to seven refer to students’ difficulties and whether these are accumulated on the theoretical/conceptual understanding of programming or/and to the application of theory in practical problems. In particular, question five concentrates the most divergent views: 40% of the teachers support that students can write workable code even though they have not grasped the theory while a 50% of the teachers support the opposite. However, there was a unanimous agreement that the understanding of the theoretical framework of programming does not indicate that students will not experience difficulties when they encounter practical problems. Finally, question four is concerned with whether teachers feel that more research is needed on the didactics of programming. Teachers unanimously agreed that more research is needed on how to approach and teach core concepts and constructs in computer programming.

4.1.2 Potential threshold concepts

The third section of the first questionnaire referred to threshold concepts and asked the participants the following question: “Based on your experience, can you suggest three or more potential threshold concepts in the broad area of function? Please provide a short description for each of the threshold concepts you suggest.”

Round 1 resulted in 27 suggestions, which were qualitatively analysed. The aim of the analysis was the generation of a list with the potential threshold concepts proposed by the
panel. For each suggested concept, a suitable entry was made in the list. However, there were some cases that were more challenging than others. For example, some participants provided only descriptions of the concept rather than single words. This created two problems: Firstly, many of these descriptions matched another concept already proposed by another participant. Thus, I had to put together concepts that were articulated slightly differently but covering the same theme. For example, a participant used the following sentence to describe a potential threshold concept: "students find difficult to understand that the code jumps around and not executed sequentially". This description was matched to the concept of Control Flow, which was suggested by another participant. I only grouped together the participants’ responses referring exactly to the same computing concept. Similar concepts like arguments and parameters were not grouped together as they have a slightly different definition in the curriculum. Secondly, some participants included more than one potential threshold concept in one description. For example, the following phrase was used by one participant, which led me to create two potential threshold concepts: “The use of argument passing and return values to reduce the need for global variables. Students initially struggle to understand the benefit of local variables over global, particularly as they may perceive the use of argument passing and return values more time consuming and challenging than simply using global variables.” The list (table 4.2) was distributed to the participants to initiate Round 2. Using the description provided for each of these concepts, a list of 19 potential thresholds was pieced together.
<table>
<thead>
<tr>
<th>Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments</td>
</tr>
<tr>
<td>Calling a function/procedure</td>
</tr>
<tr>
<td>Classes</td>
</tr>
<tr>
<td>Control flow</td>
</tr>
<tr>
<td>Converting a procedure to function and vice versa</td>
</tr>
<tr>
<td>Difference between Functions and Procedures</td>
</tr>
<tr>
<td>Difference between Parameters and Arguments</td>
</tr>
<tr>
<td>Handling Files</td>
</tr>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>Parameter passing</td>
</tr>
<tr>
<td>Abstraction</td>
</tr>
<tr>
<td>Procedural Decomposition</td>
</tr>
<tr>
<td>Recursion</td>
</tr>
<tr>
<td>Return values</td>
</tr>
<tr>
<td>Using the right sequence of function calls to solve a problem</td>
</tr>
<tr>
<td>Understanding why functions/procedures are useful and are needed in programming</td>
</tr>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>Variable scope</td>
</tr>
<tr>
<td>Global and Local Variables in functions</td>
</tr>
<tr>
<td>Visual representations</td>
</tr>
</tbody>
</table>
4.2 Delphi - Round 2

With the list of 19 concepts, the participants were asked to indicate the level of agreement on a five-point Likert scale of whether each concept is a potential threshold one. For each concept, the following statistical information was calculated: mean, median, mode, SD, IQR, and the percentage of "agree" and "strongly agree" responses. To reach consensus, the following criteria were considered for each concept: a. The percentage of the participants stated that "agree" or "strongly agree" that a concept is a threshold must be more than 75%, b. the standard deviation (SD) should be less than 1.5, and, c. the Interquartile Range (IQR) should be less than or equal to 1. These criteria were adapted from the study of Giannarou and Zervas (2014). It should be noted that 75% agreement level was chosen to increase the confidence in the study’s results and is one of the strictest in the literature (e.g. Giannarou and Zervas (2014) adopted a 51% of agreement level).

The statistical analysis followed resulted in ten concepts reaching a consensus of being threshold concepts in round two. Table 4.3 depicts these concepts.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Mean</th>
<th>Median</th>
<th>Mode</th>
<th>% agree and strongly agree</th>
<th>SD</th>
<th>IQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments</td>
<td>4.20</td>
<td>4</td>
<td>4</td>
<td>90</td>
<td>0.919</td>
<td>1</td>
</tr>
<tr>
<td>Calling a function</td>
<td>4.30</td>
<td>4.50</td>
<td>4 and 5</td>
<td>90</td>
<td>0.949</td>
<td>1</td>
</tr>
<tr>
<td>Control flow</td>
<td>4.20</td>
<td>4</td>
<td>4 and 5</td>
<td>80</td>
<td>0.789</td>
<td>1</td>
</tr>
<tr>
<td>Parameters</td>
<td>4.20</td>
<td>4</td>
<td>4</td>
<td>90</td>
<td>0.919</td>
<td>1</td>
</tr>
<tr>
<td>Parameters passing</td>
<td>4.20</td>
<td>4</td>
<td>4</td>
<td>90</td>
<td>0.632</td>
<td>1</td>
</tr>
</tbody>
</table>
In this round, the participants were asked again to review the results of the previous round and indicate a new level of agreement if they wanted to change their previous one. Three participants in total made changes to their previous level of agreement. These changes led to another concept reaching a consensus of being a threshold one, that of Abstraction, making the total number of threshold concepts eleven. Dajani et al. (1979) argue that to terminate the Delphi process, stability must be reached for each argument. Thus, to measure stability for each concept, the Wilcoxon signed-ranks test was employed as suggested by the literature (e.g. De Vet et al., 2005). By employing this test, researchers can determine whether a difference between the data of two Delphi rounds has statistical significance, thereby testing for stability of the data.

All concepts reached stability (the significance level was set at .05) and, thus, the Delphi was terminated with three rounds. Table 4.4 depicts the concepts that have reached consensus in this round. The p-value of the Wilcoxon signed-ranks test is also provided for each concept. Changes in some values in comparison with Table 4.3 indicate a change in the participants’ opinion.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Mean</th>
<th>SD</th>
<th>N</th>
<th>T</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedural Decomposition</td>
<td>3.90</td>
<td>4</td>
<td>4</td>
<td>90</td>
<td>1.101</td>
</tr>
<tr>
<td>Recursion</td>
<td>4.40</td>
<td>4.50</td>
<td>5</td>
<td>90</td>
<td>0.699</td>
</tr>
<tr>
<td>Return values</td>
<td>4.30</td>
<td>4</td>
<td>4</td>
<td>100</td>
<td>0.483</td>
</tr>
<tr>
<td>Variable</td>
<td>4</td>
<td>4</td>
<td>4 and 5</td>
<td>80</td>
<td>1.247</td>
</tr>
<tr>
<td>Variable scope</td>
<td>4.40</td>
<td>4</td>
<td>4</td>
<td>100</td>
<td>0.516</td>
</tr>
</tbody>
</table>

4.3 Delphi - Round 3

In this round, the participants were asked again to review the results of the previous round and indicate a new level of agreement if they wanted to change their previous one. Three participants in total made changes to their previous level of agreement. These changes led to another concept reaching a consensus of being a threshold one, that of Abstraction, making the total number of threshold concepts eleven. Dajani et al. (1979) argue that to terminate the Delphi process, stability must be reached for each argument. Thus, to measure stability for each concept, the Wilcoxon signed-ranks test was employed as suggested by the literature (e.g. De Vet et al., 2005). By employing this test, researchers can determine whether a difference between the data of two Delphi rounds has statistical significance, thereby testing for stability of the data.

All concepts reached stability (the significance level was set at .05) and, thus, the Delphi was terminated with three rounds. Table 4.4 depicts the concepts that have reached consensus in this round. The p-value of the Wilcoxon signed-ranks test is also provided for each concept. Changes in some values in comparison with Table 4.3 indicate a change in the participants’ opinion.
### Table 4.4 Descriptive statistics on potential threshold concepts on Round 3

<table>
<thead>
<tr>
<th>Concept</th>
<th>Mean</th>
<th>Median</th>
<th>Mode</th>
<th>% agree and strongly agree</th>
<th>SD</th>
<th>IQR</th>
<th>Wilcoxon signed Ranks test: p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments</td>
<td>4.40</td>
<td>4</td>
<td>4</td>
<td>100</td>
<td>0.516</td>
<td>1</td>
<td>.317</td>
</tr>
<tr>
<td>Calling a function</td>
<td>4.30</td>
<td>4.50</td>
<td>4 and 5</td>
<td>90</td>
<td>0.949</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Control flow</td>
<td>4.20</td>
<td>4</td>
<td>4 and 5</td>
<td>80</td>
<td>0.789</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Parameters</td>
<td>4.20</td>
<td>4</td>
<td>4</td>
<td>90</td>
<td>0.919</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Parameters passing</td>
<td>4.20</td>
<td>4</td>
<td>4</td>
<td>90</td>
<td>0.632</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Procedural Decomposition and Design</td>
<td>3.90</td>
<td>4</td>
<td>4</td>
<td>90</td>
<td>1.101</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Recursion</td>
<td>4.40</td>
<td>4.50</td>
<td>5</td>
<td>90</td>
<td>0.699</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Return values</td>
<td>4.30</td>
<td>4</td>
<td>4</td>
<td>100</td>
<td>0.483</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Variable</td>
<td>4</td>
<td>4</td>
<td>4 and 5</td>
<td>80</td>
<td>1.247</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Variable scope</td>
<td>4.40</td>
<td>4</td>
<td>4</td>
<td>100</td>
<td>0.516</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Abstraction</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>80</td>
<td>0.667</td>
<td>1</td>
<td>.317</td>
</tr>
</tbody>
</table>

### 4.4 Conclusion

This chapter presented the results of the first phase of the research. The results suggest that 11 concepts in the area of functions are considered as potential threshold concepts by experienced computing teachers. In total, three rounds were needed to reach consensus and stability. The results of this phase were used as input for the second phase of the research, the results of which are described in chapter 6.
Chapter 5: Discussion of Phase 1 – The Delphi study

The first phase of the study aimed to explore potential threshold concepts in functions in computer programming, focusing specifically on experienced computing teachers’ perspectives. To that end, this phase focused on answering the first research question, which was a more general question about teachers’ perspectives on students’ difficulties in computer programming and the usefulness of the threshold concepts framework in secondary computer programming. The second question concerned teachers’ perspectives on potential threshold concepts related to functions. The following sub-section discusses the results of the first research question, and the next one discusses the results of the second research question presented in the previous chapter.

5.1 Teachers’ perspectives on students’ difficulties in computer programming and on the threshold concept framework

Searching for the source of computer programming difficulty is not a new endeavour. Since the mid-1980s, many researchers have centred their attention on identifying students’ misconceptions. For example, Perkin and Martin (Perkins & Martin, 1986) identified four types of fragile knowledge relevant to computer programming: partial knowledge, inert knowledge, misplaced knowledge, and conglomerated knowledge. Du Boulay (1986) also identified specific areas that students find difficult and grouped these areas into categories: orientation, notional machine, notation, structures, and pragmatics. In this phase of the study, I focused on two broader categories of knowledge: theoretical and practical knowledge, or

---

7 The discussion is also part of the following paper which can be found in the Appendix: Kallia, M. and Sentance, S., (2017). Computing Teachers’ Perspectives on Threshold Concepts: Functions and Procedural Abstraction. In Proceedings of the 12th Workshop on Primary and Secondary Computing Education (pp. 15-24). ACM
declarative and procedural as was defined by McGill and Vollet (1997) and discussed in the literature review. These two categories can be seen as a distinction between threshold concepts and procedural thresholds when the threshold framework is considered.

Interestingly, teachers’ views on the relationship between the theoretical and practical understanding of programming do not converge (section 4.1.1 – table 4.1). Their responses surely indicate that the theoretical understanding of programming is not enough to ensure that students will not experience difficulties while writing programs. On the other hand, their perspectives on whether a partial or total lack of theoretical understanding obstructs students’ practical knowledge do not converge, highlighting the difference between the theoretical knowledge students need to understand and the skills they need to practice. It is reasonable that the theoretical understanding must precede students’ engagement with practice. However, there are research studies that indicate that students can write workable programs even though the misconceptions they hold have not been resolved. An illustration of such a study is the one conducted by Madison and Gifford (1997).

However, it was not a surprise to see that the participants unanimously agreed that more research is needed on the didactics of programming. Computer programming is a relatively new area in England’s secondary computing curriculum, and teachers feel that more support is needed in teaching this subject. This is in line with the research conducted by Sentance and Czizmadia (2016) in the UK and with initiatives that support computing teachers and provide training opportunities such as Computing at School (UK) and CS for All (US).

Regarding the threshold concept framework, from examining the teachers’ responses depicted in Table 4.1 (Chapter 4, Section 4.1.1), it can be argued that the framework of threshold concepts can further inspire secondary teachers’ practices; most of the participants
indicated that some of the students’ difficulties in programming could be examined and explained under the view of threshold concepts. Most importantly, a high percentage of them reported that simply knowing that a concept is a threshold concept would influence their approach in teaching that concept. This was quite a surprising result considering that there are no practical guidelines to date on how threshold concepts should be approached in teaching in any discipline. However, it would be interesting for further research to investigate teachers’ views on this matter and how they would change their teaching methods to teach a threshold concept.

5.1.1 Teachers’ suggestions on potential threshold concepts within functions

Many years of research have revealed that students experience many challenges when they start using functions and procedures. That is why most of the concepts identified in this study as threshold concepts do not come as a surprise to a reader familiar with relevant research. Fleury (1991) and Madison and Gifford (1997) reported that the concepts of arguments, parameters, and parameter passing are concepts that cause trouble in students’ understanding. In her paper, Fleury (1991) discusses constructed rules (false assumptions) that lead to students’ misconceptions around this area. Among them, she refers to parameters, values, and variable scope. Madison and Gifford (1997), in their study on parameter passing, indicate that students can write workable programs despite their existing misconceptions. They also refer to parameter passing as the most challenging concept in introductory programming. More recent studies also highlight the problems students encounter with actual and formal parameters and the order of parameter evaluation and a function call (Chen et al., 2012; Sirkia, 2012), while Miller et al. (2015) suggest that parameter passing is a potential threshold concept.
Under the lens of the thesis theoretical framework, the results of this phase based on teachers’ suggestions lean towards categorising these concepts as potential threshold concepts (table 4.4). I will highlight here the word “potential” as the participants, in their descriptions, focused mostly on the conceptual difficulty that these concepts impose on students, emphasising their troublesome characteristics and without considering one of the most important characteristics of thresholds, which is their transformative nature. However, there were cases where the participants grouped these concepts together, highlighting the connection between them and providing evidence of their integrative characteristic. For instance, one participant discussed parameter passing as follows: "Passing values into the parameters of the procedures and functions is often hard for students to get to grips with, particularly when passing variables as arguments."

Other participants linked the concept of parameter passing with variable scope and return values: "Students initially struggle to understand the benefit of local variables over global, particularly as they may perceive the use of argument passing and return values more time consuming and challenging than simply using global variables" and "Students find difficult to understand the use of parameter passing instead of global variables." These examples are evidence of the relationship of these concepts and the possible transformation that a student exhibits once these concepts are understood. A participant’s description aptly summarises this: "These concepts are the cornerstone of understanding how functions work."

The order in which functions or procedures are executed has also been reported in the literature as a source of difficulty for students. For instance, Sleeman et al. (1988) reported that students believe that the execution of functions begins when the program starts. Ragonis and Ben-Ari (2005) further reported that students believe that the order of the methods in the
program determines the order in which they are called. Sirkia (2012) found that students misunderstood how a function call works, reporting on several problems students experience.

In this study, the participants perceived calling a function and control flow as threshold concepts (table 4.4). Most of the participants again focused on the troublesome characteristic. For example, one participant stated, "The first threshold concept for students to understand is that the function will not do anything until it is called. Most students struggle to understand why it will not run when you have defined it. They assume that once they have defined a function and compiled it, that the code will execute." Another participant reported that control flow and calling a function are interrelated concepts, and once one of them is understood, the other becomes clearer, highlighting in this way their integrative characteristic.

Another area in which the literature has shown a variety of students’ misconceptions is the notion of variables in programming. Variables are undoubtedly a very important concept in computer programming and understanding them is imperative to progress in the discipline. To my knowledge, this is the first study that proposes variables as a threshold concept. The troublesome characteristic was evident in the participants’ descriptions with one participant stating, "Students can struggle with the fact that variables can store values that can be referenced in other parts of the program and that the value stored can change," while another participant highlighted its significance as follows: "I think this is a very important concept that goes beyond procedural abstraction." These examples emphasise the importance of understanding variables and indicate the impact that misunderstanding may have on students’ progress in the discipline. This concept was also mentioned many times in
the participants’ descriptions of other concepts, especially the concepts of variable scope and return values. This is probably because the concept of variables is a fundamental concept that is connected with others across programming and conceptually evolves as students move forward in the discipline.

The participants also suggested recursion as a threshold concept. Gal-Ezer and Harel (1998) propose recursion as one of the most challenging concepts for teaching. Using a phenomenographic approach, Booth (1993) recognised three different ways that students experience recursion: as a programming construct, as a means for repetition, and as a self-reference. Many studies put recursion in their focus, and a good summary of them is described in the study of McCauley et al. (2015). In their paper, they summarise years of findings of students’ challenges, mental models developed when practising recursion, and best practices for teaching this concept. Due to its compound difficulty, some studies have already added this concept to the threshold concept repository (Boustedt et al., 2007; Holloway et al., 2010; Rountree & Rountree, 2009). Along the same lines, abstraction as a threshold concept has already been proposed by some researchers (Boustedt et al., 2007; Eckerdal et al., 2006). It is indeed a core concept and, as Detienne (1997) contends, a necessary ability for object-oriented programming.

Finally, procedural decomposition is a concept that has not been reported in the literature as either a source of misconceptions or a threshold concept. However, in the study of Goldman et al. (2008), procedural design and functional decomposition were among the ten concepts with the highest rankings in terms of importance and difficulty in programming. Undoubtedly, students face difficulties in understanding how to break a problem into smaller parts and then create the corresponding procedures or functions. A recent study conducted
by Thomas et al. (2017) suggests software design as a threshold skill. Along the same lines, it seems more accurate to regard procedural decomposition and design as a threshold skill than as a threshold concept. This argument was also suggested by one of the participants that had initially suggested, "I think that this is more a threshold skill than a threshold concept," as well as indirectly by another participant who noted his/her disagreement about procedural decomposition being a threshold concept: "I was at odds with the group on decomposition and design. For me, this is a complex activity that relies on a range of knowledge, experience, and skills."

5.2 Concluding remarks

The chapter presented here discussed the results of the first phase of the research. The aim was to explore teachers’ perspectives about the threshold concept framework and to collect potential threshold concepts in the area of function. The teachers agreed on 11 concepts in functions that are threshold concepts (arguments, calling a function, control flow, parameters, parameter passing, procedural decomposition and design, recursion, return values, variable, variable scope, and abstraction) but their focus has mostly been on students’ difficulties with these concepts and not that much on students’ qualitative transformations once they understand these concepts.

Comparing the results of this study with the studies conducted in higher education, it seems that the findings suggest some new potential threshold concepts in the area of functions while also corroborate the findings of other researchers regarding the concepts of parameter passing (Miller et al., 2015), abstraction (Bustedt et al., 2007; Eckerdal et al., 2006), and recursion (Bustedt et al., 2007; Holloway et al., 2010; Rountree & Rountree, 2009).
To further collect evidence of the transformative and integrative properties of this set of concepts, the second phase of the research was conducted to uncover these properties. However, in this investigation, I did not include the following concepts: variable scope (because not all teachers had experience with global variables), variables (as this is not part of the function curriculum), and recursion and abstraction (as these have already been identified as threshold concepts by other researchers).

The results of this phase used as an input for the second phase of the research, the results of which are presented in the next chapter.
6 **Chapter 6: Results of Phase 2 – IPA study**

This chapter aims to present the results of the IPA study conducted with computing teachers in response to research questions 3 (*How do computing teachers experience the potential threshold concepts through their teaching and students’ engagement with these?*) and 4 (*Is there evidence that supports the nomination of these concepts as threshold concepts, skills and conceptions?*). This phase was critical for getting a deeper understanding of teachers’ experiences with the concepts identified as thresholds in the previous phase and for unfolding the characteristics that make these concepts thresholds from the teachers’ perspectives and meaning making of their experiences teaching these concepts. Because IPA includes an idiographic approach to research, I will first provide a case by case description of the data collected from the teachers. I will then present the results of the multicase analysis, and the findings resulted from examining the cases together.

6.1 **Teachers’ experiences case by case**

In this subsection, based on the idiographic nature of IPA, a narrative is provided for all the four computing teachers focusing specifically on how these teachers experience the practice of programming in the area of functions. In doing so, the readers will have a more holistic view of the cases that participated in the study. The names of the teachers were codified as Rea, Olivia, Andrea, and Mateo. The narratives are presented descriptively and as closely as possible to the participants’ true data.
6.1.1 Case Description - Rea

Rea is working as a computing teacher in the UK for more than seven years while she is teaching computer science at key stage 4/5 for more than four years. She has also experience working outside a school environment where she practised programming for one to three years. When she was asked about students’ difficulty in programming, she noted that these difficulties stem both from the conceptual understanding of programming and the practical work it involves. She is of the opinion that students can write workable code even though they have not grasped the theory behind the concepts or the constructs they use, but she also emphasises that the theoretical-conceptual understanding of programming components is not enough for students’ successful practice. That is why she believes that more research and guidance are needed in the area of the pedagogy of computer programming. Threshold concepts was a familiar framework for her, and, she argues that under this theoretical spectrum, many of the students’ difficulties in programming can be explained. Thus, she came in the interview with this presupposition in mind.

In the area of functions, her teaching practice focuses on the concepts of parameters, arguments, parameter passing, return values and calling a function which indicates that she places equal attention to most of the concepts that this area involves. Among the concepts that she finds most difficult for students are the concepts of parameters and arguments. She explains that students struggle to understand how these variables in the function definition are taking the values of the arguments. The key problem for her is that students fail to understand what exactly is happening when a call to a function is being made and how the arguments are going to pass the values to the corresponding parameters. Another problem
that she has experienced with her students is with the number of arguments in the calling statement. The teacher here explains that students find it quite difficult to handle more than one argument in the calling statement. Thus, some common mistakes that her students usually make are in the syntax of the calling statement, e.g. omission of the comma between the arguments or the omission of parentheses.

In her teaching practice, she associates students’ difficulties with their expressed emotions; she argues that she can understand when the students are struggling with something or the moment they have finally understood it by their expressed emotions. For example, she recalls her students while struggling to understand how parameter passing works, saying something like “oh miss, we have no clue of what is going on” or “it is difficult, I don’t understand” while other students are saying “that’s fine” and they rush through it. For her, this difference between students’ understanding depends on the individual abilities of the students.

To Rea, everything starts to make more sense after students have understood parameter passing. She explains that as soon as this happens, students see differently how all these concepts work together, how a function is being called and how the control flow goes to the function definition and returns back again. So, for the teacher, this concept’s understanding ties together most of the other concepts in the area of functions. What is interesting, though, is that she notices changes in the way that her students articulate their thoughts in programming exercises after grasping this concept. Specifically, the students make use of the right computing vocabulary, which for Rea, is an indicator of understanding. She also notices that by understanding parameter passing her students’ conceptual understanding of variables is changed and takes an additional meaning while they also better understand what
calling a function actually means and how the flow of the program “jumps” to another place. Thus, Rea sees parameter passing understanding as central to understanding functions.

About variables, the teacher considers this concept as one of the most important aspects of programming. She explains that students understand what a variable is because teachers spent much time teaching that concept. However, she highlights that the application is more difficult than understanding the concept, and this comes with practice, which helps students see the importance and the role of variables in programming. About variable scope, she notes that her students do not experience actual problems, and usually they understand the role and the use of local variables. She also argues that variable scope puts an additional meaning to the notion of variables.

To Rea, return values are not conceptually difficult for students. However, she states that students find it difficult to understand where this value will be returned and stored and how it will be handled in the main function. She has also noticed a pattern that students usually use when they have not understood this. She explains that when students use a print function and not a variable to catch the return value, this is an indicator of misunderstanding. She then tries to help the students understand why it is better to use a variable to hold the return value, and she further works on this with the students. Having understood that, the teacher argues that her students feel more confident with functions and programming, which is also evident in the way they use the computing language.

Finally, the teacher sees decomposition both as a concept that needs a theoretical understanding and a skill that needs to be practised. However, she argues that she has witnessed many of her students being able to do the coding behind this, but not being able
to explain why they did what they did while others experience the opposite problem. This, she argues, demonstrates the complexity of this concept.

6.1.2 Case Description – Andrea

Andrea has taught computing in UK schools at key stage 4/5 for more than seven years. She has also experience in practising programming for more than four years outside school settings. The teacher’s long teaching experience made her speak with confidence about students’ difficulties in programming. Regarding these, she believes that the difficulties are accumulating more in the practical application of programming rather than the theory behind the concepts or constructs of programming. Thus, she believes that students, even though they have grasped the theoretical knowledge of programming, still encounter difficulties in the practical level. She advocates that theoretical understanding does not support practice, and this was evident throughout the teacher’s interview. She believes, however, that theoretical understanding is a prerequisite and a necessity for students’ practice, but this does not indicate success in the latter. This is the reason that the teacher thinks that more research is needed in the way that core concepts and constructs in programming are being taught.

Regarding the threshold concept framework, the teacher believes that it is a valid framework and can explain some of the students’ difficulties in programming. What is also interesting for this teacher, is that she would change the way she approaches a concept if she knew that this is a threshold one. Thus, Andrea came in the interview with these presuppositions in mind.

It is very interesting to note how Andrea connects her teaching experience with the concepts identified as potential threshold concepts in the previous phase. From the start of her
participation, I could see that as a teacher she places much focus on the practical side of programming as she believes that this is the place that most of the students’ difficulties are gathered. This is also explained by her suggestions for overcoming students’ conceptual difficulties, which are based on practice and students’ exposure to graphical environments in which they can see the execution of the program in real-time visualisations. As a teacher, she thinks of programming in terms of both theoretical understandings and practical exercise. She notes, however, that theoretical understanding does not indicate students’ ability to program. For her, this is a different skill that comes with practice.

Central to her teaching approach to functions, is the role of parameter and parameter passing. The teacher puts these concepts at the top of the most difficult concepts in the area of functions. Although the teacher mentioned this explicitly, it was also evident from the fact that the teacher dedicated half of the interview’s time talking about the difficulties around these concepts and the changes that their understanding brings to students. Specifically, she thinks of parameters and arguments as one concept, and with parameter passing, she creates a conceptual area that students find it extremely difficult to cope with.

When she was specifically asked about parameters, she emphasised the concept’s difficulty for students. She states that she has experienced students struggling to understand this concept and how it works. She also highlighted that even students, who usually have not previously experienced difficulties with other concepts in programming, have difficulties with this concept. She believes that students, through practice, can better and quicker understand this concept and normally students with two to three lessons and lots of practice will start to make sense of parameters. To Andrea, students’ difficulty stems from their inability to understand from where these variables take their values or as the teacher says,
“where these variables come from?”. It is obvious for her that the problem lies in how the arguments in the calling statement give their values to the parameters through parameter passing. She explains that even though the students already use predefined functions in which they pass the arguments correctly, they fail to see the connection between what they are doing in these predefined functions and what they should do in their own functions. Thus, as the teacher explains, “that analogy does not work for students at the beginning”. The teacher further explains that in predefined functions, the students only deal with concrete things like what arguments to pass to the calling function and do not have to think about more abstract notions as the parameter.

Andrea sees parameter passing as another concept that is difficult for students. However, she believes that it is somehow more understandable than parameters. This is because, as the teacher notes, students have already grasped and understood what parameters and arguments are and, thus, it is easier to understand how parameter passing actually works. Students can understand that when calling a function, they need to pass the arguments for parameter passing to work and for parameters to get their values. Thus, the teacher places an important role in parameters for students’ understanding of other concepts such as parameter passing.

Another problematic area that the teacher highlights is the control flow of the program and students’ understanding of it. As the teacher argues, what is interested here is that students do actually understand what happens to the flow when a call is being made: “the students understand that the execution jumps around to a different place”. What students fail to see, she explains, is that during the call of the function, the values of the arguments are being transferred and allocated to the parameters. Thus, the teacher here concludes that it is difficult for students to take into consideration all these things that happen as part of calling
a function which includes a change in the flow of the program, transfer of the arguments’ value to the parameters and later on the return value.

To overcome these difficulties, the teacher suggests that students must be involved in practical exercises and particularly exercises that represent the flow of the program visually and what is happening when a call to a function is being made: “it is faster and there you can see because the flow is shown, the commands are highlighted as it goes. Moreover, this way the control flow is very visible, because there’s also some of the activities they’re using some subroutines so you can see how it jumps to subroutine, how it’s repeated and then how it’s returned.” During these exercises, the teacher argues that differentiating the values of the parameters and seeing how this change affects the outcome visually (e.g. the construction of a triangle), helps students’ understanding of parameters.

To Andrea, what is also important for students’ understanding of parameters and parameter passing, is the ability of students to abstract or think abstractly. The teacher here notes that the age of students is not of importance as she has witnessed seven-year students not experiencing problems with parameters and parameter passing while older students, e.g. 10-year ones, experiencing serious problems. The teacher contributes this to the students’ ability to think abstractly but also to the students’ ability in Maths. She states that she expects students that have understood some algebraic principles to get quite well with variables and then with parameters. The teacher sees functions and variables as a different kind of abstraction and highlights that students’ lack of abstract thinking prohibits learning and working at this level.

For Andrea, being able to write workable programs does not indicate an understanding of what exactly the program does. Thus, the teacher says that she had experience students being
able to write workable code, but when they were asked questions about their code, the students were not able to give plausible answers. Additionally, they cannot correct errors by themselves and, thus, when an error arises, they cannot go through this without the teacher’s help. Error correction is central to the teacher, and it is an indicator of understanding. Pattern recognition is also very difficult for students as they cannot understand or point out key features in a case or when some features are patterns of the same thing.

The teacher finds that the experience the students acquire as a result of grasping both the notion of parameters and parameter passing are the ones that influence most the students’ thinking in functions and therefore in programming. What is interesting, though, is that the teacher emphasised the transformation that the students exhibit when they finally understand these concepts. Specifically, the teacher sees a change in students’ way of thinking about real-life problems and scenarios that involve information processing. They also start seeing that computation is not just something that happens only with a computer. Thus, the teacher argues that the understanding of these concepts enhances students’ higher-order skills. For example, she marks that they use abstraction in more efficient ways than before and they also start working with problems having in mind the effectiveness and efficiency of their programs and alternative ways and solutions to decompose the problem.

Another change that students experience because of this is their notion of the flow of the program. The teacher here argues that students’ theoretical knowledge about the execution of the program and how this jumps around with a call to a function, is now improved as the “theoretical knowledge alone does not support understanding”. Not surprisingly, though, the teacher emphasises that the immediate effect she can notice is on students’ practical work, which becomes better. In addition to that, she also highlights that she can observe
students correcting their own errors without the help of the teacher. As a final remark, the teacher can observe a change in the way the students use computing language, but she notes that this is an outcome which comes after the practical work. In the end, the teacher finds that the students’ confidence in programming is also enhanced as soon as the students’ understanding of these concepts is enhanced.

To Andrea, the notion of variables is changed, or an additional meaning is attributed to variables as a result of understanding parameters. Actually, the notion of variables is extended as an additional role is added depending on the context, she argues. Regarding the concept’s difficulty, the teacher comments that students generally do understand variables in a simple situation, but in a real-life situation which they have not worked before, they find it difficult to understand which variables to use. It is the same with functions which are another type of abstraction: as soon as the students encounter a real-life scenario, they find it difficult to think of what functions to use. Thus, she notes “it is hard for students to transfer a real-life scenario into a computable version”. The variable concept is a very important concept for her and she highlights that by saying: “The notion of a variable in general it is very important so once a person got it a lot of other things make much more sense, and they’re able to solve a wider range of tasks, yes that’s for sure”.

The teacher, however, does not think that the variable scope is difficult for students to grasp. She further mentions that as soon as students understand parameters and parameter passing and calling a function, then the variable scope is quite easy for students. When the teacher was asked about the return values, she said that this is also a very important aspect of functions. Students usually do not quite understand why this is needed and why a value should be returned. It does not make sense to them why a function must return something
back even though they have worked with predefined functions, but they do not make the connections and are confused. What makes sense to students at this point, is to describe the need for the return values, based on the variable scope and local variables that are used in the function. However, the teacher admits that usually, the students are taught the return values before the scope of variables. She also notes that after some practical work, the students can finally understand why return values are needed.

When the teacher was asked about any changes that occur on students after they understand return values, she admitted that it is hard to notice a change other than the improvement of students’ practical work. However, she did state that when they do understand that, they better see the concept of calling a function and what really involves as well as how the thing that was calculated in the function now is being used and handled in the main function as a result of the return statement. She also highlights the connection that students see with calling a function and return values.

Finally, the teacher was asked about decomposition. She did argue that decomposition is really difficult for students, especially when they are dealing with more complicated scenarios and the reason for that is because it involves higher order skills. She also thinks that decomposition involves both a theoretical understanding and a practical skill but the theory, she notes, it is not enough to help students’ practice. Having understood that and been able to practice decomposition, the teacher finds that students consider the value of this process: “They understand the value, how it works, and how it helps and affects the practical side of work”.

The teacher generally highlights that in all these concepts discussed, there are skills that need to be practised except for the theoretical understanding of the concepts. It is also evident that
the teacher places greater value to the parameters and parameter passing as the most difficult concepts in functions and the ones that yield an important transformation in students.

6.1.3 Case Description - Olivia

Olivia is working as a computing teacher for many years and has experience at key stage 4 and 5 computing curriculum for more than seven years. Her experience in programming also includes one to three years of practice outside school settings. Her experience as a teacher has led her to the conclusion that students’ difficulties in this discipline stem from both the subject’s conceptual difficulty and the application of theory to practical problems. As such, she believes that students’ conceptual understanding does not necessarily indicate their ability to generate workable programs. At the same time, however, she notes that students can generally write workable code even though they have not grasped the theory behind constructs and concepts. On the framework of threshold concepts, she advocates its efficiency to explain students’ difficulties in programming while she believes that generally, teachers need more research studies on how to teach this subject.

A central theme in her teaching is the way that students use the language to describe a phenomenon. She argues that this is an indicator of students’ understanding and a way to instil understanding to students. That is why her teaching practice has a focus on learning and understanding and using the right vocabulary in programming. As a teacher, she pays equal attention to all the concepts that surround the “function” part of computing. Most difficult from her perspective, though, are two concepts: parameters and parameter passing. She argues that to understand these concepts, students’ perception of a subroutine must be
complete. What is interesting though is that she sees this as a skill that students can establish when they reach year 9.

When she was asked about the source of students’ difficulties in this specific part of the curriculum, the teacher mentioned that students struggle to understand how arguments are passed to parameters in the function definition but also students find it difficult to understand when there is a need for more than one argument and parameter. She sees these difficulties in many of her students, but she highlights that some students already think in a computationally way and, thus, they see functions, parameters and parameter passing as a natural thing in programming. To Olivia, students need to develop abstract and mathematical thinking to understand these concepts. However, instead of that, she places an important role in students’ motivation for learning something. Without this, she argues, it is very difficult for them to understand anything deeply.

The teacher sees as an additional problem the language that is being used in computing. She is an advocate of students’ using the right vocabulary to describe things in programming. What is interesting, though, is that she finds that students are often intimidated by the vocabulary of computing rather than the phenomenon that is represented by it. Another problem that she has noticed is students’ difficulty in comprehending how a program “jumps” from place to place. She finds that students’ familiarity with a program that is executed line by line makes more difficult the understanding of that a program can jump from place to place. In other words, the teacher considers the flow and what this is comprised of troublesome for students and its understanding must precede students’ engagement with functions and parameter passing.
The teacher suggests that trying to explain this with real-life examples is really helpful for students as well as the visual representation of the flow of the program. The teacher also suggests using trace tables, get students to look at the pace and structure diagrams that let them work through what is happening in a program without just running it. She argues that if they just run the program and it works, the students do not learn much from that, emphasising the significant role of practice. From her experience, most students need six to seven months to completely master these concepts and be able to use them by themselves. Having understood parameters, and parameter passing, the teacher, believes that students see programming and subroutines differently. She argues that students understand the usability of decomposition and that they can write code that they can re-use. They understand what an efficient piece of code looks like and here is when the flair comes in which she believes that it gets developed through their understanding of subroutines and passing parameters and arguments. Apart from that, she highlights that students become more confident and competent in using functions, and they use computing language correctly. The teacher also notes that they become much better on reading and tracing the code as if they are able to compile it in their heads, which denote an understanding of the program’s flow.

When she was asked about variables, she highlighted their importance in understanding programming in general. The teacher also emphasised the difficulty that students experience when they use variables as arguments instead of just values. She explains that naming and what to put in the arguments list and the parameters list is of importance and students often are confused with that. Especially with the use of variables, which is a more abstract concept, the students need a bigger understanding of how the program works to manage that. She also
notes that students’ understanding of variables is changed through their involvement with parameters.

When the teacher was asked about local variables, variable scope, she said that students do not find difficult how these variables work. However, she emphasised that students’ understanding of return values is changed after that because they understand that since these variables cannot be accessed outside the function, something must be returned back. Moreover, with return values, the teacher argues that it is initially difficult for students to understand them, but that also depends on the way that the teacher approaches this concept. Also, the way that the teacher connects problems with real-life examples, and the practice with structure diagrams influence students’ understanding of this concept. Having grasped that, the teacher has experienced students appreciate better what is happening to the data and how different parts of a program communicate while they also start feeling more confident in programming and understand the role of functions as something that has a purpose.

For her, structured diagrams and design are central to students’ understanding of decomposition and functions. Decomposition is something that students also find tricky at the beginning. The teacher believes that students must find a useful example to work on initially, to understand the value of decomposition, which is a challenge for teachers. To Olivia, decomposition’s difficulty lies both in the concept and the skill behind this, but the value of decomposition can only be realised through practice. Thus, the teacher believes that decomposition and its value comes after practice: students should spend some time on writing programs which they have first broken down into smaller parts and brought it back together as one program. That is where they get the concept and the skill of decomposition rather than in learning the theory behind it. Having understood that, the teacher argues that
students see programming differently. “That’s where their flair comes in”, the teacher said. They think about turning their focus from the output and internal workings to efficiency, “It boosts their ego”.

6.1.4 Case Description - Mateo

Mateo is a computing teacher with many years of teaching experience. The variety of teaching experience includes both students at different key stages but also teachers who are trained to be computing teachers. Concerning students’ difficulty in programming, the teacher emphasises both the conceptual difficulty of programming, the time and practice the students dedicate to it but also the teaching practices that the computing teachers adapt. He is a strong advocator against the use of analogies that have been proven not to help students understanding but to obstruct learning. Threshold concepts are a familiar framework to him, although he did not mention whether he supports the existence of threshold concepts in programming or not. Thus, this teacher came to the interview with no previous formed preconceptions about the threshold concept framework.

His approach to functions first concentrates in letting students realise why functions are useful. That is why he likes letting students write many lines of similar code just for them to realise or ask the question: “can we do this with fewer lines?” For him, that means that they are ready to understand the role of functions and, thus, he can proceed to this part of the curriculum. In any case, he believes that students must work hard and need a lot of practice to get the grips of functions.

As a teacher, he places equal attention to all the concepts that functions include. However, he recognises that students’ difficulties in this area are mostly concentrated around the
concept of parameters and parameter passing. Among the various sources of students’ difficulties, he finds that the type of computer language and the notion of variables are the most significant ones. For example, he notes that using the box analogy to explain what a variable is, causes many problems to students, problems that are then transferred to the area of functions. Another problem that he recognises and attributes to teachers is that they are teaching students that the arguments and parameters must have the same name, which leads to students experiencing many difficulties when they start coding by themselves. Another conceptual difficulty that the students encounter refers to the way that the arguments are transferring the values to the corresponding parameters. The teacher argues that this is initially very difficult for students to comprehend. When he was asked if value tables are something that he considers helpful for students, he noted that he did not think so because students do not see the connections between the program and the tables.

The teacher strongly believes that when students finally understand what a parameter is and how parameter passing works, there is a transformation that takes place in students’ minds. He argues that he can see this understanding from the quality of students’ programs. Students make shorter and more efficient programs while the teacher also notices a change in the way they communicate their ideas using the right vocabulary. Mateo also argues that in Python, students start seeing that everything is an Object and, thus, they better understand object-oriented programming.

Variables are something that the teacher sees as central to programming and a source of several misconceptions that influence understanding of functions. He further notes that global variables are something that students find difficult to understand, especially when global variables should be used and how. The teacher explains that students tend to overuse
these variables is not a good practice. He argues that students need a lot of practice to be able to judge when to use these variables and when not to. An interesting point that the teacher also made was that students tend to use global variables to avoid return statements. He explains that this is a common mistake students initially make and teachers need a lot of work and planning to help students resolve this misconception. Having completely understood how local and global variables work, the teacher states that students can actually understand the need for return values, and how functions generally work.

To Mateo, return values as a concept is not so difficult for students. However, he highlights that students are struggling to understand how the return values are handled in the main function. Here, the teacher notes that students experience difficulties with the syntax and the variable that stores the return value. When the teacher was asked if he has noticed any difference in students’ attitudes after grasping this concept, he said that students could understand the power that their program could have and how functions actually work. He also has noticed students feeling more competent and confident in handling functions and in programming in general. “This also gives them joy”, he argues. However, he noticed that not all students reach that level, and he remarks that he does not expect students to reach such a high level of transformation.

Another difficulty that students face in this area of programming is stemming from their prolonged engagement with code that is executed line by line. The teacher here argues, when students start learning functions, they experience difficulties in understanding that the flow can jump to other lines of code. What is also difficult for students is to create a comprehensible and a well-structured program. Here, the teacher emphasises that students
can write functions correctly, but sometimes they do not know how to create a coherent program with them.

In the end, the teacher was asked about decomposition, and he noted that this concept is a challenging one for students. He explains that students have the tendency to get coding straight away without planning or thinking about the steps they have got to go through before that. For him, thus, students need first to understand what decomposition is, and then they need much practice in order to understand its value. Thus, this concept needs both an initial theoretical understanding, but without practice, students cannot see the importance of decomposition in programming: “practice reinforces knowledge of why decomposition is important”. The teacher was also asked if he had noticed any changes in students after the understanding of this concept. He explained that as soon as they understand decomposition and how it works, they can break down a problem and tackle each one of the steps at a time which leads to more efficient solutions to a problem. Finally, Mateo noted that decomposition could help students’ understanding of functions and the flow of the program.

6.2 Teachers’ experiences across cases

In the previous section, I presented the teachers’ experiences case by case as a way of depicting the individual way of experiencing the potential threshold concepts in their daily practice. As I have already mentioned in the Methodology chapter, the last step of IPA and the most interesting one is to compare the cases in search of similarities and differences and with the purpose to provide an overall description of the phenomenon. Therefore, this section aims to present the findings for the whole teachers’ group and, thus, explores all the cases together as a multicase.
6.2.1 IPA Analysis and Findings

Before the interviews, I decided that I should focus on the concepts that have not been identified before as threshold concepts and are specifically part of functions. To this end, I did not ask the participants to reflect on their teaching experiences with the following concepts: abstraction, recursion, and variables. The IPA analysis of the interviews from the cases as a whole resulted in 12 prevalent themes for the concepts of parameter, parameter passing, return value and procedural decomposition and some of the themes were further clustered into superordinate themes. For the rest of the concepts (arguments, calling a function, control flow) no prevalent themes emerged and, thus, these concepts and their themes are not presented in this thesis. For the concept of “variable scope”, the specific teachers interviewed did not have experience with teaching this concept and, thus, the concept was also not further examined. According to Smith (2011), a theme with frequency at least half of the sample size or more is prevalent and, thus, I will focus on these themes that are most compelling to address the research questions of the study. The following table depicts the themes for each of the four concepts that the interview was focused on. The table also includes quantitative information and specifically the number of participants that mentioned each of the 12 themes.
Table 6.1 Themes from all the cases

<table>
<thead>
<tr>
<th>Superordinate and subordinate Themes</th>
<th>Number of participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>1 Difficulties</td>
<td></td>
</tr>
<tr>
<td>Parameters are conceptually difficult</td>
<td>2</td>
</tr>
<tr>
<td>Skills needed to understand parameters</td>
<td>2</td>
</tr>
<tr>
<td>2 Impact on self or learning</td>
<td></td>
</tr>
<tr>
<td>Parameters and Conceptual changes or coherence</td>
<td>3</td>
</tr>
<tr>
<td>Parameter Passing</td>
<td></td>
</tr>
<tr>
<td>1 Difficulties</td>
<td></td>
</tr>
<tr>
<td>Parameter passing is difficult for students</td>
<td>3</td>
</tr>
<tr>
<td>Overcoming parameter passing difficulties</td>
<td>2</td>
</tr>
<tr>
<td>2 Impact on self or learning</td>
<td></td>
</tr>
<tr>
<td>Parameter passing and Personal Transformation</td>
<td>2</td>
</tr>
<tr>
<td>Parameter passing and Conceptual changes or coherence</td>
<td>3</td>
</tr>
<tr>
<td>Return values</td>
<td></td>
</tr>
<tr>
<td>1. Difficulties</td>
<td></td>
</tr>
<tr>
<td>Return values are conceptually difficult</td>
<td>4</td>
</tr>
<tr>
<td>2. Impact on self or learning</td>
<td></td>
</tr>
<tr>
<td>Return values and changes in students’ understandings</td>
<td>4</td>
</tr>
<tr>
<td>Parameters, Parameter Passing and Return values</td>
<td></td>
</tr>
<tr>
<td>1. Impact on self or learning</td>
<td></td>
</tr>
<tr>
<td>Students’ Transformations</td>
<td>4</td>
</tr>
<tr>
<td>Procedural Decomposition</td>
<td></td>
</tr>
<tr>
<td>1. Difficulties</td>
<td></td>
</tr>
</tbody>
</table>
Conceptual and Practical difficulties | 4
---|---
2. Impact on self or learning | 
The value of decomposition and students’ flair for programming | 3

### 6.2.2 Concept 1: Parameters

Parameters are concepts that have been discussed in the literature as troublesome or as concepts that cause misconceptions on students’ learning. That is probably the reason why, from all the nine concepts discussed in the interviews, parameters were among the concepts that the teachers most frequently referred to. Table 6.2 depicts the prevalent themes of this concept.

**Table 6.2 Themes for parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Theme Frequency (number of participants)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Difficulties</td>
<td></td>
</tr>
</tbody>
</table>
Parameters are conceptually difficult | 2 |
| | Skills needed to understand parameters | 2 |
| 2. Impact on self or learning | 
Parameters and Conceptual changes or coherence | 3 |

### 6.2.2.1 Parameters are conceptually difficult

All the four teachers agreed that students experience many difficulties in understanding the concept of a parameter and most importantly, how parameters and arguments are related. In
other words, most of the teachers’ points focused on how parameters take their values from the arguments which will be discussed in the concept of parameter passing.

When Andrea was asked about what is difficult with understanding parameters, she replied:

“With parameters I think because when for example we use Python for GCSE and say we are looking at how you can define a function, and they don’t understand so we use the names of the variables to know the parameters and they think where these variables come from? So is it something that I’m already using in my program or...? So they don’t quite get that idea even though they use predefined functions and they pass their arguments into predefined functions, they cannot maybe link what they’re doing with the functions that they are in Python to the functions that they’re doing themselves. So that parallel kind of, that analogy it doesn’t work at the beginning”. (Andrea, Line: 31-36)

Here, Andrea explains that students experience problems with understanding how to decide and define the corresponding parameters for the functions they create. She notes that even if students can use correctly predefined functions and pass the arguments appropriately, they seem not to be able to do the same with their functions. What the teacher tries to say here, is perhaps that in predefined functions the students only deal with concrete things like what arguments to pass to the calling function and do not have to think about more abstract notions as the parameter.

Yet, for Olivia, another source of the problem seems to be the language. She explains that students, in general, have problems understanding the vocabulary of computing, and this is a deterrent factor for the conceptual understanding of the subject:
“I don't know. I think if you strip it all... I think part of the problem is the language. I think in order for them to understand they need the vocabulary first. And I think that that sometimes is the barrier, it's not the fundamental understanding of a parameter or an argument, it's the language that makes it sound too scary. So if you strip that away, the language or if you develop an understanding of the vocabulary I think it becomes much easier to understand. I think with a lot of these concepts that that's the big issue, is the language”. (Olivia, line: 33-37)

The teacher’s observation about the difficulties that the vocabulary imposes on students reveals the multifaceted difficulties that they encounter in programming. Indeed, the teacher explains that it is not just understanding of what a parameter is or an argument is, but students may, in the beginning, be intimidated by the new vocabulary that they are introduced to. Being afraid to engage with something unfamiliar to their own discourse experiences may prohibit their learning.

6.2.2.2 Skills needed to understand parameters

Two teachers referred to skills that may be needed to help students’ understanding with parameters. Andrea specifically mentions that the age of students seems not to affect their understanding as there are some younger students that can cope with these concepts while others, older ones, cannot. She advocates that students’ level of abstract thinking is probably one reason of why some students get it quicker than others:

“So I think it probably depends on... Not on their age but maybe on the way they can deal with abstract notions like variables. So, for example, some Year 7’s don’t have any problems with that but some people even in Year 10 will still have a problem.”.... And it’s quite
connected to their understanding of Maths. For example, normally you would expect if a student is in a top Maths set they probably won’t have as many problems as someone who is in I don’t know set three or four or something like that. I guess normally if they understand like some algebraic principles then they get on quite well with variables. (Andrea, line: 87-90)

The teacher also refers to students’ ability in Maths. She explains that usually, students that have an understanding of some algebraic principles get on quite well with variables and, thus, with parameters. For her, functions and variables represent a different kind of abstraction, and she highlights that students’ lack of abstract thinking prohibits learning and working at this level.

For overcoming some of the above difficulties, Andrea emphasises the role of visual environments:

“... and because in these visual environments they can use these things without maybe really fully understanding. But then through the usage they... I think because they see so many different examples, they form some kind of idea themselves of what it is and how it works..... You can also, they also allow you to define parameters like for example when you do a function or a procedure so draw a rectangle... A triangle. And you can just do one without parameters, and then you can add a parameter and see how it changes, how different triangles go out and so it’s just some exercises that is probably easier for them to understand then when they code, like in Python for example”. (Andrea, line:8-14 and 69-77)

In this extract, Andrea explains that with visual environments, students can work on many different examples. Their engagement with these will eventually help them understand
parameters and arguments. However, what is striking in the extract is that the teacher understands that students need a different level of complexity on their tasks. She explains that it is better for students to start working with functions with no parameters and gradually adding these to their programs. In this way, students could see the difference in the output and understand what is changed when they add one or more parameters and most importantly realise the parameter’s role in functions. Thus, visual environments can help students understanding of parameters and arguments and how these communicate or how data are transferred through the program. Nevertheless, students need time, commitment and practice to really understand what is going on.

Following Andrea’s thinking, Olivia emphasises students’ computational thinking with understanding parameters and arguments.

“And for some students, you know, it really is a very difficult concept but for some that naturally kind of think in a computational way it's not very difficult at all... Their mathematical thinking wasn't there yet, so it was very difficult. Whether that could be done eventually over time I'm not sure”. (Olivia, line: 26-29, 130-132)

Olivia advocates the computational thinking role in understanding these concepts. She has noticed some students not encountering any problems with these concepts, and she attributes that in the way they are thinking and specifically, in computational thinking. For others, she continues, these concepts are difficult, and she attributes that to students’ lack of mathematical thinking which obstructs students’ understanding. Therefore, for Olivia, computational and mathematical thinking are important and a determining factor for students’ understanding.
When teachers were asked about the changes that occur on students once they understand parameters, Rea, Andrea and Olivia commonly agreed that students better realise the notion of variables or that the notion of variables is enhanced or further extended:

“For example, the lesson I have taught them, I explained about parameters and variable and then they actually understood that in the function definition we can give it any name, like I can call it X,Y but when I am calling it I may be passing a value 5 and 6, so X goes 5 and Y goes 6, that’s the whole concept of those variables, the parameters in the function are actually caching those values and then you can manipulate them as whatever way you like in your call. So yeah, that definitely consolidates the concept of what is a variable”. (Rea, line:77-83)

The way the passage unravels strongly suggests the strong connection that Rea makes between variables and parameters. She starts first by noting that it is important in teaching to highlight the connection between parameters and variables. She explains that once this is comprehensible by students, they can see that parameters can have any name, as variables do, but in contrast with variables, parameters’ values are not assigned to them by the assignment operator. Instead, their values are assigned to them “indirectly” from the calling statement, and specifically from the arguments. As such, students can then understand that parameters can take different values that could potentially change the outcome of the function. Looking at the following extract, Olivia seems to have been engaged in a similar experience with her students:
“Yes I think so. Particularly if you're selling the idea of creating a subroutine, which can be used in like if you're making a subroutine that does a specific job but it's a flexible enough that the fact that a variable was being passed in means you'll have a different result depending on what's passed in, I think that's something that they learn. And I think that's a big lightbulb for them”. (Olivia, line:167-170)

Olivia emphasises the function’s flexibility and role. She explains that students’ understandings of parameters influence their understandings of variables. Once students make the connection between parameters and variables, in other words, when they understand the role of parameters as variables, they can see and understand the function’s role. What is obvious to them is that changing the parameters’ values causes a potential change in the function’s output.

Andrea also refers to variables as a concept that is actually extended with students’ engagement and understanding of parameters and arguments. She explains that by saying:

“Yes it makes their... Because now the notion of variable is extended because they can see this is not just something maybe we use them in the main program, it might have some different scopes and different roles depending on the context, yes. So that certainly improves the variable understanding”. (Andrea, line:178-181)

Andrea’s extract captures the transformation or the extension of students’ knowledge about variables. As she explains, students can see that variables are not only used in the main function of a program and consequently on calculations, but they can also have other roles like that of a parameter or an argument.
6.2.3 Concept 2: Parameter passing

The concept of parameter passing was also one of the concepts that the teachers reported most frequently during the interviews. The teachers discussed both the concept’s difficulties and the ways to overcome these problems. They also referred to students’ transformations that capture both conceptual changes, integration of knowledge and personal changes as well. Table 5.3 depicts the prevalent themes for this concept.

<table>
<thead>
<tr>
<th>Parameter Passing</th>
<th>Theme Frequency (number of participants)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Themes</td>
<td></td>
</tr>
<tr>
<td>1 Difficulties</td>
<td></td>
</tr>
<tr>
<td>Parameter passing is difficult for students</td>
<td>3</td>
</tr>
<tr>
<td>Overcoming parameter passing difficulties</td>
<td>2</td>
</tr>
<tr>
<td>2 Impact on self or learning</td>
<td></td>
</tr>
<tr>
<td>Parameter passing and Personal Transformation</td>
<td>2</td>
</tr>
<tr>
<td>Parameter passing and Conceptual changes and coherence</td>
<td>3</td>
</tr>
</tbody>
</table>

6.2.3.1 Parameter passing is difficult for students

Three of the participants clearly explained the reasons for parameter passing difficulty on students. Rea neatly captured these difficulties when she says:

“I think it is difficult for students. Yes, the difficulty lies on how these variables in the function definition are going to take the values from the arguments. So yes, they can call a function, but they don’t fully understand of how arguments are being handled. The key concept is
when you’re calling a function, and you are passing certain parameters in your function call, how are they being held in the function definition that’s the key thing that students struggle to understand”. (Rea, Line: 3-6)

In the first sentence, Rea emphasises the relationship between arguments and parameters and the conceptual difficulty that this relationship imposes on students. Rea underlines the conceptual and not the procedural difficulty as she clearly demonstrates that students can call a function correctly and pass the corresponding arguments but how these arguments are being handled is still something that students struggle to understand. Thus, for Rea, the difficulty with understanding parameter passing focuses on students’ endeavour to comprehend how the arguments in the calling statement are transferred in the function definition and are stored in the corresponding parameters.

Another problem was also mentioned by Rea. She explains that her students have problems with the number of arguments in the calling statement and, thus, they find it quite difficult to handle more than one argument in the calling statement. Some common mistakes that her students usually make are in the syntax of the calling statement, e.g. omission of the comma between the arguments or the omission of the parentheses:

“I think, for instance, when the function is called with a single argument or multiple. That’s something that they don’t fully understand. So what they are struggling with is they don’t like, if I am just talking about syntax of python, they don’t separate it with the comma, if they are trying to pass two arguments, they just don’t separate them and then a classic mistake is that when they call a function they just forget to use the brackets. It’s not that they have not understood it, it’s like they are rushing through it. These are the normal syntax errors they do”. (Rea, line:30-35)
A similar experience was demonstrated by Olivia. She emphasises students’ difficulties to deal with more than one parameter by saying:

“And I think there's a difference between just passing one parameter in and passing multiple parameters. I think that's kind of the next level up from understanding that data can be passed in.” (Olivia, line: 23-24)

Drawing on her teaching experience, Olivia also emphasises another problem: naming variables, arguments, and more generally naming identifiers. She explains that students are confused when they have to pass variables as arguments because this requires a more abstract way of thinking than passing simple values. She demonstrates that by saying:

“I think sometimes it's naming. So naming... The difference between naming things in the main program and what gets passed in and understanding what's in it through the identifiers that they use. And sometimes... See because we don't use it very much in terms of just calling a specific function and passing in a data value, we more use it with passing in variables... So that itself is difficult but I think if they put in a specific value that they call a subroutine and they put in the value that's going to be used, I think they understand that much better than if they pass in a variable. Because that's more abstract, isn't it? That takes a bigger understanding of the whole program” (Olivia, line: 77-85)

The almost same opinion is expressed by Mateo, who argues that teachers make the mistake of teaching students, at the beginning of the course, that parameters and arguments should have the same name. This teaching approach confuses students as later, in the course, they learn that this is not actually true:
“Well I think again a lot of it comes down to the person who’s doing the teaching not knowing the difference between a parameter and an argument. And at an early stage I think that many people, many teachers are teaching students the argument and the parameter have to have the same name, which isn’t true. So when they start to read code on, whether it’s on Stack Overflow or on something that’s written specifically for students, that they begin to actually find quite difficult…..So it’s the idea that the function can take an object, which doesn’t have the same name as the parameter and the function would then handle that, that can be quite difficult”. (Mateo, line: 63-72)

Olivia also reflected on another source of difficulty for students. She describes that there is a leap the students have to make to understand how parameter passing works. From Olivia’s extract, it is obvious that the teacher refers to the flow of the program and the understanding that in programming, some lines of code are executed sequentially while others are not. According to her, students find it hard to make this transition which the understanding of parameter passing definitely requires and it would be helpful, she argues, if teachers use real-life examples to demonstrate this property. Finally, she places the role of computing language among the main factors that may prohibit students from understanding this concept:

“I think from when they go from a basic program, I think it's the leap, the difference. They can understand, you know, if I had a set of instructions if I do X, Y, Z then that's executed in that order. The difference between sort of instructions that are out of line, I guess they'd understand it better if they had a real-life application for it. .. I think if you just try and explain to them in the programming language it's hard for them to grasp but if you begin to explain to them with a sort of real-life example I think they begin to understand it a bit better.
And again, I think part of it is language also and I think what doesn't help with that is that different examples have different vocabulary and even that changes”. (Olivia, line: 45-54)

6.2.3.2 Overcoming parameter passing difficulties

Suggestions for overcoming students’ difficulties were made by two teachers. Using their experience of what has worked for their students, they moved on describing their teaching approach. The key point of their discussion was visual representation tools and tools that demonstrate the flow of the program. As such, both these teachers think that understanding how the program is executed is really important and vital for overcoming the difficulties above:

“I think so. I think it depends how you teach it because I think if you get them to look at how it all fits together, like for example if you get them to look at [unclear] execute cycle and you get them to think about the program counter for example and you get them to think about that's sequential isn't it? Unless you have a break in which case it jumps to a different point in the program...If you use something like this from a computer and they can see how it is executed, I think it helps to build up a whole picture of everything. I think things in isolation are more difficult to understand than a deeper understanding that stems from [unclear] execute cycle”. (Olivia, lines: 65-73)

Here, Olivia reports that students’ transition from a program that is executed line by line to a program that jumps from place to place is difficult for them. Letting students understand the cycle of execution and how the values of variables are changed as the program is executed, it is vital for Olivia. She further emphasises that by using a visual environment to illustrate this would be much better for students’ understanding. Additionally, she suggests
that trace tables and structure diagrams help students work through what is happening in the program before actually running it, and this is important for understanding this concept. She also firmly believes in the role of vocabulary for enhancing students’ understanding and suggests that students should be engaged with it and not be afraid of it:

“Vocabulary. Understanding of vocabulary and not being afraid of it. Trace tables and get them to look at the pace, to manually look at the pass of data through a program. Structure diagrams. Just formal structures that let them work through what’s happening in a program without just running it. Because if they just run it and it works I don’t think they learn very much from that”. (Olivia, line: 87-92)

A similar argument is made by Andrea. From her experience, she has noticed that it is helpful for students to be engaged in exercises that visually demonstrate the flow of the program and the commands that are executed as the program is running.

“I notice that sometimes it helps... There is, you probably... You’ve been code.org... and they have various, little activities for coding different types of structures and we use in our school we use accelerated course, it’s like a 20 hour course but normally you can do it faster and there you can see because the flow is shown, the commands are highlighted as it goes. ... And this way the control flow is very visible, because there’s also some of the activities they’re using some subroutines so you can see how it jumps to subroutine, how it’s repeated and then how it’s returned”. (Andrea, line: 60-67)
6.2.3.3 Parameter passing and Personal Transformation

When teachers were asked about changes that occur on students once they understand parameter passing, two teachers referred to personal transformations. Specifically, Rea refers to students perceived levels of confidence and their competence in programming:

“Yea, they definitely feel more confident. Because the way we start in teaching this like you give them ... and they try, but once they have understood that concept everything starts to have more sense for them. And that's like a key theme, oh yes they actually trying to understand of how a function is called and the control goes to where the function definition is and all those instructions are executed and then it returns back to the main”. (Rea, line: 40-44)

She explains that once students understand parameter passing and how it works, they start feeling more confident and competent in programming. She emphasises that once they grasp this concept, everything begins to make more sense for them, and specifically the way functions work and how they are being called. Olivia too underlines students’ improved ability in programming by saying that she can see their competence and confidence be enhanced when students’ attempt to work independently without asking for the teacher’s guidance:

“You always get that lightbulb moment, which I think is the sort of thing you're looking into, where they get it and they can use it independently”. (Olivia, line: 127)

Yet, for Rea, there is another change that takes place. She explains that understanding parameter passing causes a shift in the way students articulate their ideas in programming. In other words, once students grasp this concept, a transformation is taking place, which is
aptly demonstrated in the way they use computing vocabulary. The students start articulating their ideas in a more scientific way:

“Yea, some people who have fully understood they can say the word correctly but some students who would not previously be able to articulate but then they were like “oh miss what you pass in there”, what is going in there, and I was like what do you mean by what is going in there. They weren’t using the right vocabulary but they were referring to what was going on”. (Rea, line: 67-71)

6.2.3.4 Parameter passing and Conceptual change and Coherence

The teachers also reflected on changes that occur on students on a conceptual level once they grasp the concept of parameter passing. The teachers specifically refer to variables, parameters and arguments, control flow, the concept of a function and object-oriented programming. Specifically, Rea makes most of the above connections. She refers to variables, parameters and argument but also on the flow of the program:

“I think again understanding of variable is like a container. That’s the key concept that they get straight away. It’s the integration of what they have learnt previously and then they can developed a link of why it works. (1, 88) ... Parameters and arguments take another conceptual shape as soon as they understand how parameter passing works. And also, the main concept is, they also understand that when you call a function what exactly happens. So, yea. The flow of the program makes more sense to them. I have heard, I can recall one of the students passing a comment “oh yeah now I get it”. That’s was really pleasing. They are grasping a concept in more depth”. (Rea, lines: 94-97)
There is much to analyse in this extract. Rea explains that parameter passing enhances students’ understanding of variables. She refers to an integration of knowledge taking place where the students can connect what they already know with their new knowledge. Rea also notes that once students grasp how parameter passing works and what exactly is happening during this process, they start seeing the connection between parameters and arguments and how these communicate. At the end of the transcript, the teacher also explains that the flow of the program and apparently the non-sequential execution of the code starts making more sense to the students. For Rea, it may be that students begin to understand how data are transferred through the code and how different parts of the program communicate with each other.

Olivia also refers to students’ understanding of functions and their role in programming once parameter passing becomes clear to students:

“I think just the whole idea of subroutines to be honest. Subroutines, yes. Functions and procedures” (Olivia, lines: 165-166).

Finally, while Mateo also demonstrates the same experience with Olivia and Andrea about parameter passing and functions, she extends this experience further by explaining that at least in Python, students start to understand that everything is an object, and this facilitates their understanding of object-oriented programming:

“Still thinking in terms of variables so I think it begins to reinforce the idea that in Python everything is an object. So once they’re clearer about that, I think the [unclear] function works is clearer and therefore when they come onto tackle object-oriented programming,
which is probably in Year 12 Key Stage 5, things are clearer for them”. (Mateo, lines: 275-282)

6.2.4 Concept 3: Return Values

Return values were also among the concepts that the teachers discussed during the interviews. The discussion about this concept mostly concentrated on the difficulties on students’ understanding and less on changes, both personal or conceptual that students’ experience once they understood this concept. The following table depicts the prevalent themes for this concept.

Table 6.4 Themes for return values

<table>
<thead>
<tr>
<th>Themes</th>
<th>Frequency (number of participants)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Difficulties</td>
<td></td>
</tr>
<tr>
<td>Return values are conceptually difficult</td>
<td>4</td>
</tr>
<tr>
<td>2 Impact on self or learning</td>
<td></td>
</tr>
<tr>
<td>Return values and changes in students’ understandings</td>
<td>4</td>
</tr>
</tbody>
</table>

6.2.4.1 Return values are conceptually difficult

All the participants discussed the difficulty that return values cause to students. It is interesting that the reasons they report, capture a wide range and do not only focus on one aspect of this concept.
Looking at an extract from Rea, it is obvious that the teacher focuses more on how students are handling the function’s return value in the main function (or in the point that it is returned to). She explains that students have difficulties in understanding why the return value must be stored in a variable. Rea also explains that students’ way of handling this demonstrates if they have understood return values or not. For example, if students use a print statement to print the function’s outcome instead of using a variable first to store the outcome, the teacher explains that this is evidence that students have not yet completely grasped the notion of return values.

“If they get the concept of the function call and arguments and everything, what is difficult for them to understand is that when you return something that needs to be caught in something. That’s really important. They are using a variable to catch it usually in where they call the function. Sometimes they just send it to a print function and if they do that they don’t fully understand it, so I try to enforce the concept that catch it into a variable, so when a function returns a value that is caught into a variable and then you can print that variable to see what is the return value”. (Rea, lines:135:141)

There is almost the same opinion expressed by Mateo, although he concentrates mostly on the syntax of getting the value from the function:

“It’s the syntax of how you get the value from the function that’s difficult”. (Mateo, line: 515)

Yet, for Andrea, students’ difficulties lie on a deeper level of understanding. She clarifies that students fail to see the reason why a function should return something back and even
though when they use predefined functions, it seems that they have captured this idea when they turn to their functions they are still confused with this concept.

“I think that’s quite important, yes because they sometimes they don’t understand in the initial stages that why is it something needed to be returned. Because normally they’re used to working with just one program and nothing... It doesn’t work like something is returned and even so they use the functions like for example random or something like this they do return values and they use these values when they start creating their own they still get confused”. (Andrea, lines: 252-257).

From a different angle, Olivia concentrates more on the teaching approach used to explain this concept to students. She advocates the use of real examples, like a factory, to demonstrate the need and use of returning something back. Contrariwise, she explains that if teachers are too abstract from the beginning, this will encumber students’ confusion:

“I think it depends on how you teach it. I think if you too abstract too quickly they panic a bit and therefore that panic stops them from seeing the bigger picture but if you get them to role model. I don't know, a factory where you do a particular job and then you pass that finished part of that job back to the main bit of the factory, they understand that. And I think it's that development of that almost like a coaching from concrete to abstract so eventually they get to the point where they can do it themselves, I think that's the important part”. (Olivia, lines: 261-268)
6.2.4.2 Return values and changes in students’ understandings

Evidence of changes in students’ understanding of programming was reported by all the teachers. These are mostly concerned with the data being transferred across the program, the calling statement and the variable that stores the return value.

When Rea was asked about changes in students’ understanding having grasped the concept of return values, she replied by saying that students understand where the return values are returned to and what variable to use to catch this value. This was exactly the problem that Rea mentioned when she was asked about the difficulties students’ experience with return values: students’ have difficulties in understanding how to handle the return value in the main program. As a result, the only direct effect she can see on her students’ understanding is that they surpass this problem.

“But they do understand that when a function call is returned and from where you are calling the return values return there, so they can interpret what the values are and what variable they can use in the call successfully”. (Rea, lines: 160-162)

The same opinion was addressed by Andrea. She explains that students’ understanding of calling a function is enhanced as well as the relationship between return values and what is going on in the main program. In other words, students understand how the return values are being used and handled in the main program.

“Yes so with return values I think maybe they understand better the call of the function so they can see how that affects what’s happening in the main program, how it can be used, how to calculate or produce some results”. (Andrea, lines: 293-295)
However, Andrea clarifies that it is difficult to document students’ conceptual changes when they understand this concept. She argues that this is not so obvious as with the previous concepts.

“They might because it’s hard to... So their understanding related to the changes after the return values it’s not coming through that obviously so it’s quite hard to see how that effects... Obviously if they for example their programs are functioning correctly, say they can do the test one, they can discuss it, then you can see that they understood how it works but how it affected their other thinking about other structures and other concepts, that’s hard to judge”. (Andrea, lines: 274-281)

Finally, there is one more conceptual change that Mateo and Olivia have experienced with their students. Mateo explains that understanding return values completes students’ understanding of what a function can actually do while Olivia emphasises the same opinion by saying that students can understand the purpose of a function:

“You can do all those things and then return a validated name for a function so I think they begin to see actually functions can do a lot. Can take something and actually carry out a number of tests and then give back something which is sensible and usable from nonsense which has been put in”. (Mateo, lines: 543-546)

“Yes I guess it's easier for them to understand functions because they can see that it has a purpose that returns something. Less easy for them to understand a procedure”. (Olivia, lines: 328-330)
6.2.5 Group: Parameter, parameter passing and return values

It is quite interesting that, often, during the interviews, the teachers referred to the concepts of parameters, parameter passing and return values as a group of concepts. Specifically, when they were asked about changes in students’ personal level, they grouped together these three concepts to report on these changes. That is possibly the reason why for the concepts discussed earlier, there is not enough evidence of students’ personal transformation once they grasp these concepts. However, when these three concepts are grouped together, they seem to cause a strong transformation on students as recorded by all the teachers. The following table demonstrates the prevalent themes of these web of concepts:

<table>
<thead>
<tr>
<th>Parameters, Parameter Passing and Return values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Themes</td>
</tr>
<tr>
<td>1. Impact on self or learning</td>
</tr>
<tr>
<td>Students’ Transformations</td>
</tr>
</tbody>
</table>

While all teachers demonstrate the transformation of students once these concepts are understood, there are similarities but also differences in how this transformation is portrayed. Most of the teachers think that these concepts’ understandings and the interaction among them are essential to students’ competence in programming and in understanding the role of functions and what they can do with them in their programs.
“It’s difficult for me to explain. Once they understand all these, parameter passing, returns, parameters, arguments then it’s something like clicked into their minds and understand what’s going on”. [Rea, lines: 156-157]

Looking at Rea’s extract, it is obvious the importance that the teacher gives to these concepts with regard to students’ understandings of functions. She explains that it is not clear what exactly is going on in the students’ minds that makes them understand things that previously were not clear to them. However, she notes that it is like something clicked into students’ minds, and they understand better what is happening.

Andrea aptly describes the students’ personal transformation by saying:

“I think with parameters and parameter passing they can maybe look at practical problems in a slightly different way and they probably even can think of like real-life scenarios. So maybe a different length because we work with some real-life examples quite a lot just because some analogies are quite helpful for a lot of children I think it’s important to understand that computation is not just something that happens with a computer. And I think they’ll look at other problems that involve information processing, which is probably changing.” [Andrea, lines: 137-143]

The way the extract unfolds strongly suggests that Andrea has experienced a change in her students’ way of thinking as a result of understanding parameters and parameter passing. This is reflected from the beginning of the transcript by saying that students are thinking about practical problems in a different way, while also considering real-life problems. This suggests that students start thinking about how programming is applied to everyday problems. They start to see that “computation” is not something that happens from the
computer itself but is something that starts first by their thinking processes which are then “transferred” or expressed in a way that computers can process. This is indeed a very strong transformative experience that Andrea demonstrates in this transcript.

Andrea also explains that as a result of fully comprehending parameters and parameter passing, the students’ practical work becomes better. Students also understand when there is a need to create a function while also can correct their errors by themselves. Both of these demonstrate a deeper level of understanding:

“I think in a lot of people, maybe the first change that you see is their practical work coming out better. So, because they understand a bit more how to use it and when they write their other functions and their new tasks, then they can correct the errors. They can understand how they’re progressing better. I think practical change will be probably first and then if their practical work is successful then they can explain it better as well so verbally or in writing... I think it generally improves their higher order skills. They can use abstraction in maybe more relevant, more efficient ways like they’re looking like at the real-life problem. They can identify what parameters are needed for a particular subroutine or how it’s going to help to make the solution more effective and efficient. So, I think that’s all how they need to decompose the problem using various functions.” [Andrea, lines: 159-163 and 170-175]

The teacher further continues by saying that by understanding these concepts, students’ higher-order skills are enhanced. They can think more abstractly and also start considering the effectiveness and efficiency of their programs. Taking all together, Andrea’s experience teaching these concepts demonstrate that students can locate and correct errors in programming, to think abstractly, to seek more effective and efficient solutions and to connect their experience in class with real-life problems. Surely, all these transformations
reveal a change - an epistemological change - on students’ way of seeing programming and themselves in this course.

Almost the same arguments were mentioned by the other teachers as well. For instance, Olivia talks about an improvement in students’ skills and specifically in decomposing a problem and in understanding that the individual pieces of code can be reused which makes their programs more flexible. She also argues, as Andrea did, that students’ understanding of the efficiency of the code is enhanced. For her, this is the moment when students’ confidence is increased and start developing the flair for programming. She argues that this is something that cannot be taught, but that nevertheless is still an important moment in the learning process:

“Yes I think so. When they understand the whole thing about parameters, return statements and parameter passing, they can see that they can decompose a problem down and they can write a small piece of code that they can reuse but that is very flexible. I think that’s when they really get their idea of an elegant piece of code, an efficient piece of code, I think that’s where the flair comes in. You know that flair that you can’t really teach ... and definitely boosts their confidence. I think once they’ve mastered it it’s almost like an intermediate point they’ve reached. And I think they get an enjoyment from it at that point also which ... in itself gives a bit of momentum to their understanding and their learning.” [Olivia, lines: 136-142 and 148-151]

Olivia also highlights that once students’ understanding is completed with return values, students finally understand what is happening with the data across their programs. Students can understand how the data are being transferred from one point to another in their code.
She also notes that it is important in this endeavour to explain the reason for using the right identifiers to be easier for students to understand and track what is happening in their code:

“I think it’s [change when they understand return values] the understanding of what’s happening to the data, that’s what I think. Looking at things jumping around and a key to that I think is teach them about good identifiers really early on because if you have a good identifier it kind of, you can almost read it in English if that makes sense.” [Olivia, lines: 280-283]

For Mateo, students’ competence and confidence in programming are increased once all these concepts are understood. He explains that this is depicted in the students’ program’s complexity. Students start to understand the power that their programs can have that was not previously achievable without all these concepts. He also believes that this makes students start enjoying programming while engaging with it:

“Yes, understanding all these concepts, make them feel more confident... and their functions can be more or less complex and students, the more I think the students will greatly enjoy that... I think they can then make their program, I think it can give their program greater power so the program can then begin to do things like calculating scores or validating names that it couldn’t do before.” [Mate, lines: 557-558 and 523-525]

Olivia also reported another experience in this matter. She explains that students become better in the reading of the code as they can nearly compile the code in their head. This is a huge step for students, being able to effectively read, understand and execute the code, and demonstrates a deep understanding of the way the program is executed. As the teacher says, this ability surely increases their confidence. Their confidence is also improved for another
reason, the teacher explains. She argues that students’ abilities on using the computing vocabulary become better as a result of grasping these concepts and that by itself increases students’ confidence.

“I think with all these, they get much better at the reading because they begin to almost compile it in their head if that makes sense or interpret it I guess. So they've actually run that program in their mind and once they can master that and they can jump from place to place I think that massively improves their confidence”. (Olivia, lines: 179-182)

“I think I spent a lot of Year 8 this year just teaching vocabulary and that in itself gives so much confidence and they understand when I'm trying to explain what I mean but that took a long time to get to that point but I definitely think it's worth investing that time”. (Olivia, lines: 156-158)

Finally, Andrea’s experience also made her support that the flow of the program is better understood once these concepts are understood by students.

“The flow of the program. Because they can... Once they understand all of these, yes then their understanding of the flow has improved, yes because of these as specific examples that are still their knowledge and understanding about it. Because if you don’t use functions in practice, then all the knowledge about the flow of the problem is quite theoretical. So and then children, theoretical knowledge normally doesn’t support their understanding that much” (Andrea, lines: 147-154).
6.2.6 Concept 5: Procedural Decomposition

Another concept that the teachers discussed most frequently during the interview was the concept of procedural decomposition. The teachers expressed very interesting opinions for this concept which were focused on two things: the first is whether they believe that the difficulty of this concept lies on its conceptual understanding or a skill that lies behind the concept; the second is the changes on students’ understandings or attitudes towards programming once the concept is grasped.

The following table depicts the prevalent themes for this concept.

<table>
<thead>
<tr>
<th>Procedural Decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Themes</td>
</tr>
<tr>
<td>1. Difficulties</td>
</tr>
<tr>
<td>Conceptual and Practical difficulties</td>
</tr>
<tr>
<td>2. Impact on self or learning</td>
</tr>
<tr>
<td>The value of decomposition and students’ flair for programming</td>
</tr>
</tbody>
</table>

6.2.6.1 Conceptual and Practical difficulties

When teachers were asked about this concept and specifically if its difficulty stems from its conceptual understanding or a skill that students need to master, all teachers argued that students first need to understand the concept in a more theoretical way and then they need a lot of practice to be able to use and master procedural decomposition. This is most powerfully captured in Andrea’s extract:
“It probably goes in parallel because their practice will inform their understanding and they probably cannot practice without at least like a little bit of knowledge given to them. So, they need to grow like side by side as they practice more maybe knowledge is provided and practiced again, and again, and again on a different level... With decomposition, I think it is possible to understand it but not really being able to use it.” [Andrea, lines:321-325 and 360-361]

Following Andrea’s thinking, it clearly demonstrates the skill that is needed to capture this concept. Andrea, at the beginning of the text, explains that it is, of course, natural that students need first to be theoretically introduced to this concept before start practising with it. However, later in the extract, she demonstrates the role and importance of practice and particularly on different levels of difficulty. The most interesting point of her discussion is the last sentence: in this, she plainly says that even if students understand the role of decomposition and perhaps the benefits and why it is used in programming, this does not mean that they can employ decomposition in their programs. This is a clear indication of a skill behind this concept with obstructs students’ learning and the ability to break down a problem.

A similar experience is showed by Mateo, who explains that students need first to understand why decomposition is important, and then they need to practice for reinforcing what they learn and for understanding its value:

“I think it’s a concept that needs to be understood and then it needs to be reinforced by practice. So, they’ve got to understand first of all why it’s important. I think the practice reinforces early knowledge of why it’s important.” [Mateo. 825-826]
For Rea too, students need both the conceptual understanding and the practice to fully grasp the concept.

“It’s first a concept but you can’t have the concept and not the practice, so it needs to be like a combination of both I suppose.” [Rea, 180-181]

While all participants referred to the difficulties students’ encounter with procedural decomposition, there are differences in how these are portrayed, but all highlight the skill rather than the conceptual part of the concept.

“I think it’s as all of the high order thinking, it’s really hard because it might work okay on a very simple problem, more artificial style of problem, but if it was a real-life scenario, even if it’s just a GCSE level like real-life scenario then without any help they will be really confused how to apply the skills that they’ve learned on a simple problem to this one.” [Andrea, lines:313-317]

Andrea’s extract demonstrates the teacher’s experience with procedural decomposition and the difficulties her students encounter. She explains that decomposition needs higher order skills and, thus, her students find it difficult to practice it when they face a real-life scenario. In these examples, the teacher argues that students cannot proceed without her help. Olivia seems engaged in a similar experience with her students:

“I think it is quite tricky for them at the beginning. What’s sometimes tricky is there’s not a meaningful, you can’t give them a meaningful example at the beginning for them to code because that’s beyond them... So, part of the balance is between giving them a low enough example, concrete enough example so that they can understand the programming concept while trying to keep them motivated enough on the big picture that eventually this will be
worth it if that makes sense. Because it’s not until they actually do a meaty piece of work that they actually do value the idea of breaking things down.” [Olivia, lines:341-342 and 352-356]

Olivia explains that in the beginning, students find it quite difficult to practice decomposition. One of the reasons is that they cannot practice in a meaningful example as this is quite difficult for them. Olivia’s experience reveals the challenge the teachers face to provide a low-difficulty example that can demonstrate the value of decomposition so that students can really understand its importance and how it is practised. However, she explains that students need a substantial amount of time practising to actually understand decomposition.

6.2.6.2 The value of decomposition and students’ flair for programming

When teachers were asked about the changes in students’ understandings or in the ways students see the discipline or in students’ personal skills and attitudes towards programming, three teachers reflected on their experiences and provided strong transformational evidence of this concept.

“I think once they understand it, the purpose of it and at least they’ve seen some examples, they understand the value of it. How it maybe can be used in the task that they have. So, but it’s probably amount of time because it requires a lot of practice to see how it actually works and how it helps and how it affects the practical side of work as well… As this kind of area that helps people to really move on in their understanding and I think that decomposition is on a different level of thinking because it’s like a more general type of skill and it’s quite hard with all of these.” [Andrea, lines:337-341 and 353-355]
Here, Andrea illustrates a change in students’ ways of seeing decomposition. The students start valuing its importance, and how they can use it in their programs to increase the effectiveness of their practical work. For Andrea, once decomposition is grasped, students’ understanding is progressed to a different level of thinking.

“I think that’s where their flair comes in. There’s almost a competitiveness if you like to get things to work as efficiently as possible... I think it’s a boost for their ego.” [Olivia, lines: 395-400]

The effect of Olivia’s passage vividly demonstrates the central point of this theme. The teacher explains that learning and understanding decomposition leads to students developing the flair for programming. Students learn to work and look for efficiency in their programs, but also it makes them feel better about themselves and raises their morale in programming.

Mateo also explains that students get to see what they can accomplish with decomposition. He says that students understand that once they break down a problem into parts, they can work at each of this part at a time. They do not necessarily need to work at each of the functions at the same time and tackle everything at once. Instead, they can go back and forth and refine or redesign their functions as they move along. At the end of the extract, Mateo highlights that once students really understand procedural decomposition, then functions and the control flow of the program are easier to manage:

“Well because then once they’ve broken down what the steps and stages are, they can tackle each step at a time. They don’t necessarily have to get each step completed in its entirety and perfectly because they can always go back to it but if they’ve broken down their functions, if their program is working in functions, even if each function isn’t perfect it is,
you can go back and refine it, separately from the main program. You can’t necessarily
tackle everything at once... And I think that a clear understanding, I think a clear initial
decomposition of the program will make the control flow and the functions easier to
manage.” [Mateo, lines: 831-836 and 864-866]

Furthermore, Mateo argues that students understand that it is uneconomic to repetitively use
the same lines of code to accomplish the same thing and, thus, they realise the usefulness
and effectiveness of using functions in their programs:

“And they first of all they execute one line after another and it all gets very uneconomic and
then one can say well that’s absolutely great now can we do it in fewer lines? And why would
we want to do it in fewer lines because resources are finite. Want your program to run as
fast as possible to make as little impact on the computer as possible so can we now compress
any of these lines and then we can start using functions. Because the programs I think the
programs can become shorter. I think that’s probably the major difference and they’re
beginning to avoid the repetition.” [Mateo, lines: 168-173 and 243-244]

6.3 Conclusion

This chapter presented the teachers’ experiences teaching the concepts of parameters,
parameter passing, return values and procedural decomposition. Through an interpretative
phenomenological analysis, the teachers’ experiences were depicted and presented both
through their individual accounts as well as a whole.

Reflecting on their experiences and practice, the teachers highlighted specific conceptual
and practical difficulties that their students face with these concepts. Most importantly, the
teachers were able to identify conceptual connections that students make once these concepts
are grasped, emphasising in this way the integrative aspect and nature of these concepts. Of paramount significance was the teachers’ experiences seeing their students being transformed by comprehending these concepts and by establishing the conceptual connections this understanding entails.

The analysis of the interviews revealed that indeed the concepts of parameters, parameter passing, and return values appear to have some troublesome, integrative and transformative characteristics. These characteristics become more evident and stronger when the concepts were grouped by the teachers to describe their students’ transformations. This point is further discussed in the following chapter where the teachers’ experiences are taken into consideration in order to answer the corresponding research questions of this thesis.
Chapter 7: Discussion of Phase 2 – IPA study

In this chapter, I will discuss the findings presented in the previous chapter that attempt to answer the third and fourth research questions of this study: “3. How do computing teachers experience the potential threshold concepts through their teaching and students’ engagement with these?” and “4. Is there evidence that supports the nomination of these concepts as threshold concepts, skills or conceptions?”.

The aim of this phase of the research was to record the teachers’ experiences and find evidence that would support the nomination of the suggested concepts as thresholds. Specifically, the third research question was about teachers’ experiences teaching the potential threshold concepts identified in the first phase of the study, and the fourth research question was about the existence of evidence in the teachers’ accounts that would lead me to the conclusion that the suggested concepts are threshold concepts. Of particular importance, as explained in the introduction of this thesis, are three of the five properties of threshold concepts, namely, troublesome, transformative, and integrative. I reiterate here that the bounded characteristic is valid for all the concepts proposed, as all of them refer to functions in programming, and thus, they delineate the conceptual area of functions in programming. The irreversible characteristic is also a feature that does not need a thorough exploration. As Meyer and Land (2005) point out, if a concept is transformative, that indicates that it is also irreversible.

The following subsection focuses on the third research question, while the next one addresses the fourth research question. At the end of this section, a summary is provided with some concluding remarks stemming from this phase of the research.
7.1 How do computing teachers experience the potential threshold concepts through their teaching and students’ engagement with these concepts?

During the interview, the teachers reflected on their experiences teaching the concepts of parameters, parameter passing, return values, and procedural decomposition. The teachers, based on their experience, presented important aspects of these concepts and, specifically, the difficulties that students encounter with them, as well as the changes that understanding these concepts can evoke on students’ personal level, confidence, competence, and the way they see programming or an aspect of it.

It has been suggested that the area of functions in computer programming incorporates many difficulties for students, and many studies have referred to the problems that students experience with parameters, parameter passing, and return values (e.g., Madison & Gifford, 2002; Fleury, 1991). The findings in this study echo previous studies that discuss students’ difficulties with functions. For example, one of the studies that employs the same theoretical framework with this thesis is the study of Miller et al. (2015). In this study, the authors argue that parameter passing is a threshold concept, and they base their argument mostly on the difficulties that students encounter with this concept, although encountering difficulties with a concept does not directly imply that the concept is a threshold. They discuss students’ difficulties with parameters and parameter passing, such as problems with aligning parameters and syntactical errors when calling a function. Consistent with these findings, the teachers in our study reported similar experiences with their students. They highlighted students’ problems with using more than one parameter and argument in the function call, as well as syntactical errors and using variables as arguments instead of values.
Another problem the teachers reported about parameter passing was understanding how the flow of the program is affected by the function call and understanding how the data are transferred from the calling statement to the function definition. Interestingly, a similar problem was reported in Sleeman et al.’s (1986) study. They specifically identified two common errors. The first one refers to the statements inside a procedure, which students assume are executed in the order that they appear, and the second refers to the time at which procedures are executed, which students erroneously think that this execution is happening when the procedures are encountered in a top-to-bottom scan.

In the same context, Ragonis and Ben-Ari (2005) further reported students’ difficulties in understanding where the values of parameters derive from and where the return value of a method is returned. Indeed, the teachers also reflected on the same problems with parameters and return values. Students had difficulty understanding how and from where parameters take their values, where the return value goes, and how it is handled. The teachers also mention a confusion between the return statement and the print function, which is a problem that Miller et al. (2015) also identified.

While this research does not offer any new information regarding students’ difficulties in functions, it is the first study which provides empirical data regarding the changes the students experience once these concepts are understood. These changes refer to conceptual transformations and integration of knowledge, as well as students’ personal attitudes towards programming. All teachers reflected on their experiences seeing students transformed once they understood these concepts. In the following paragraphs, I discuss these transformations along with my suppositions regarding their place in the threshold concept framework.
7.2 Is there evidence that supports the nomination of these concepts as threshold concepts, conceptions, or skills?

While evidence of transformation and integration of knowledge was apparent in each of the concepts of parameters, parameter passing, and return values, I am reluctant to suggest that each of these concepts are threshold concepts, as the evidence was not robust enough. However, when these concepts are grouped together, the transformations reported by the teachers are much stronger, and thus, I can argue that they form threshold conceptions as defined by Perkins (2007). In the following section, these transformations are discussed in detail to support the above argument.

7.2.1 Parameters, parameter passing, and return values

As was presented in the previous chapter, the teachers reflected on their experiences teaching a set of concepts in functions with a particular focus on parameters, parameter passing and return values. Starting with the concept of parameters, most of the teachers emphasised the conceptual relationship between parameters and variables, and particularly they focused on how understanding parameters enhances or changes how students see variables in programming.

With the concept of parameter passing, the teachers highlighted both conceptual changes and coherence, as well as personal transformations on students, which suggests both the concept’s integrative and transformative characteristics. In more detail, the teachers referred to students’ competence and confidence in programming being increased once students totally grasped this concept. They particularly highlighted that students feel more confident in programming once this threshold was crossed, which gradually help them start working
independently without the teacher’s help. What was even more transformative once this concept is grasped and according to the teachers’ experiences, is that students articulate their thoughts by correctly incorporating and using the computing vocabulary. What this suggests is that students’ understanding of this concept is reflected in the way they use computing vocabulary. I should note here that one characteristic of threshold concepts is that they are discursive, which means that they cause a change in the way the learner uses the discipline’s language (Harrison et al., 2014). On the integrative side of this concept, most teachers referred to connections with other concepts related to functions like parameters, variables, and control flow, as well as the role of functions in programming.

With the concept of return values, the teachers focused mostly on students’ competence in handling return values as well as conceptual changes. Particularly, the teachers reported that understanding return values and how to use them in programming enhances students’ understanding of functions and what they can actually do and achieve, as well as what calling a function really means.

It is evident that even though return values and parameters cause many conceptual difficulties for students, when taken alone they do not seem to cause a transformation in students that could lead me to the conclusion that these two concepts are threshold concepts. On the contrary, of these three concepts, parameter passing seems to hold most of the threshold concept characteristics. As I have already mentioned in the results chapter, the teachers often grouped all these concepts together and discussed their transformative and integrative characteristics. In fact, these concepts evoke stronger transformations when they create a group that includes parameters, parameter passing, and return values together. Perkins (2007) refers to such a group as threshold conceptions. Specifically, he highlights
that the difficulty and transformative feature of some concepts may not stem from the concepts themselves but rather from the way some concepts interact with each other to create an *underlying game* which causes a deep transformation on students’ understanding. It is therefore very likely that the concept of the parameter, parameter passing and return value to form a threshold conception in programming. This conception could refer to functions as these concepts are core concepts in understanding functions in programming.

Going back to the teachers’ reflections, they first mentioned that once students grasped these concepts together, their understanding of functions, their role, and how and why they are used in programming become clearer. The teachers also reported that the students finally understand how the data in their program are being transferred and, thus, how different parts could communicate with each other. As a result, apart from the expected increased quality of the students’ work, the teachers have witnessed their students start considering real-life problems and scenarios and how information processing and programming could be used to handle these problems. The code’s effectiveness and efficiency\(^8\) become part of their thinking, and their programs obtain greater power and complexity. Teachers also stated that students understand the effectiveness of decomposing a problem, reusing the same code more abstractly, and their higher order skills and abstract way of thinking are enhanced—these all are characteristics that describe computer programmers. This is the moment that students actually advance their skills in programming, as one of the teachers aptly noted. Finally, the teachers also mentioned that students increase their confidence in programming, and they start using the computing vocabulary in more appropriate and scientific ways.

\(^8\) I should note here that when teachers talk about efficiency, they refer mostly on the reusability of code, invoking the same function in different places of their program instead of re-writing the same lines of code again and again.
Taking the teachers’ experiences into consideration, overcoming the conceptual and practical difficulties of parameters, parameter passing, and return values engages students in a transformational journey where in the end, students’ understandings and knowledge are enhanced and integrated, and their way of thinking reflects that of the practitioners in this field. Therefore, it can be argued that students experience both an epistemological and an ontological transformation as they try to think like computer programmers, which leads me to the suggestion that these concepts together form a threshold conception in computer programming.

In computer programming specifically, it is very difficult to isolate a concept and be sure that all the difficulties that students’ face or, as in this thesis, all the transformations that students experience stem from a single concept. This is because computer programming is a very sequential subject, which means that one concept is very closely tied conceptually and practically with another concept. For example, to understand parameters, students must have a good understanding of variables and then extend this schema to incorporate parameters. Therefore, the conclusion about the set of parameters, parameter passing, and return values forming a threshold conception seems more accurate than suggesting that one of these concepts is a threshold concept. This is because, even though parameter passing has enough evidence to be a threshold concept when it is grouped with the other two concepts, the transformations reported are much stronger to the extent that a threshold concept or conception would entail. However, I must note here that this conclusion focuses only on the observations and experience of the four teachers that were interviewed. This does not mean that other studies would end up with the same conclusion as this study; other studies may
collect further evidence that would suggest that parameter passing is, in fact, a threshold concept by itself.

7.2.2 Procedural decomposition

Procedural decomposition was the most controversial concept of all the eleven concepts identified in the first part of this study. This is because the teachers in that study suggested that this is more a skill that needs to be practised than a concept that needs to be theoretically understood.

Particularly in computer programming, there are skills that students need to practise in order to understand programming entirely and to be able to write code at an advanced level. For example, many studies in computer programming distinguish between declarative knowledge, which refers to knowledge about concepts and principles, and procedural knowledge, which refers to the “active, strategic use of the declarative knowledge” to solve a problem (Palumbo, 1990:66; Lau & Yuen, 2009). This distinction is also evident in the threshold concept framework. For example, Davies and Mangan (2007) specifically referred to discipline and procedural thresholds: The former refers to knowledge coming into focus through the acquisition of theoretical perspectives and the latter though ways of practising. Thomas et al. (2017) use the construct of threshold skills to refer to procedural knowledge acquired by practice.

From the teachers’ interviews around this construct, the evidence found suggests that procedural decomposition is a procedural threshold or a threshold skill rather than a threshold concept. That is because the teachers, even though they highlighted the theoretical knowledge that students need to understand first, they emphasised that this is something that
is reinforced by practice and that students need a lot of practice to finally master decomposition. They also mentioned that even if students understand the role of procedural decomposition in programming theoretically, this does not imply that they can actually apply decomposition to their programs. This strongly suggests that there is a skill that needs to be practised by students to fully grasp procedural decomposition.

Specifically, the teachers’ experience with teaching decomposition reveals that they believe that there is a transformation that students undergo once they master decomposition. The teachers argue that this transformation leads students to appreciate what they actually can achieve and accomplish by employing decomposition, to start to value its role and importance in programming and to see how their programs can be more effectively and efficiently written. The teachers also mentioned that this is one of the moments where the flair for programming is developed and where the students start thinking about problems differently by applying higher-order skills.

The changes referred to here reflect the kind of transformation that threshold skills provoke as defined by Thomas et al. (2017), who argue that “mastering a threshold skill transforms what students can do and their vision of what they can do,” while also enabling students to see "other possible applications, broadening the list of tasks student can perform or enabling them to perform them in a new way” (Thomas et al., 2017:335).

Taking everything into consideration, the changes the students experience, as reported by their teachers, once they master decomposition, indicate an epistemological shift where the students start developing their thinking in a way that echoes the scientific thinking of this field. Therefore, I suggest procedural decomposition as a procedural threshold or a threshold skill in computer programming.
7.3 Concluding remarks

This phase of the study focused on teachers’ experiences in functions and particularly their experiences teaching the concepts of parameters, parameter passing, return values, and procedural decomposition. Based on the teachers’ experience, I first highlighted some common difficulties that students at key stages 4 and 5 encounter while trying to understand these concepts. This specific output is of great significance as it can be used by teachers who would like to organise their lessons around difficult points in this specific thematic area of the curriculum and prepare appropriate materials that will help students overcome the corresponding obstacles.

Secondly, the study explored students’ transformation and integration of knowledge once these concepts are grasped. This is the first study to investigate these transformations within the specific set of concepts by employing an interpretative phenomenological analysis of interviews with experienced computing teachers. The results of this analysis led me to propose procedural decomposition as a possible procedural threshold (threshold skill) and the group of parameters, parameter passing, and return values as a possible threshold conception. It is important to highlight here that the findings should be interpreted with caution. As this is an IPA study, I cannot produce an objective statement about the transformation all students will exhibit once these concepts are understood. However, I can support that these concepts appear to have strong transformational properties as evidenced by the teachers’ experiences and may be experienced in the same way or in a different transformative way by the students and teachers.
The findings of this phase could have a significant impact on the teaching of computer programming not only in secondary education but also in higher education. Teachers can be encouraged to take into consideration the powerful conceptual relationships and links that these concepts form and the transformations that students exhibit as a result of them. Creating integrated knowledge in learning environments is critical, and therefore, a potential next research goal could take this problem into consideration and explore ways to design a programming curriculum that will integrate students’ existing knowledge within and out of the computing discipline.

Additionally, the importance of transformations occurring during the learning process has long been documented by Mezirow (2000). The challenge in the current education system and for educators is how to present new information after having previously helped students to revise or transform their existing knowledge. Therefore, with some of the students’ transformations having been identified, a next research goal could also identify appropriate teaching strategies and practices that will more easily bring about these transformations and assist students in their attempt to understand and identify themselves in the computer classroom. I am particularly interested in exploring learning theories and different transformative learning and teaching strategies that could potentially develop a constructive teaching process appropriate for fostering transformative learning.
Chapter 8: Results of Phase 3a - Affective dimensions of learning and threshold concepts

The current chapter presents the data analysis and the results of the third phase of the research and particularly the first part. The aim of this phase was to investigate affective factors linked to liminality and therefore understand how students experience liminal space. The factors that are investigated are the following: self-efficacy, calibration, task value, sense of belonging, motivation and computer science identity for both students in liminal and students in the post-liminal space.

The chapter starts by presenting the interrater reliability, sample demographics, and the descriptive statistics for the examined variables and then it moves forward to presenting the results for each of the affective constructs explored in this phase of the research.

8.1 Preparation of the data

Before analysing the data, the reliability of the test administered to the students was examined. To this end, the Cronbach’s alpha was calculated in SPSS, which is the mostly used reliability coefficient. The results shown below suggests that the test employed is an acceptable measure with alpha=.789 (Tavakol & Dennick, 2011). I used Cronbach’s alpha based on standardised items because the questions had different metrics (Falk & Savalei, 2011). In the following tables, the variables q1 to q6 refer to the corresponding programming exercises of the test.
### Table 8.0.0 Case Processing Summary

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cases</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valid</td>
<td>123</td>
<td>100.0</td>
</tr>
<tr>
<td>Excluded</td>
<td>0</td>
<td>.0</td>
</tr>
<tr>
<td>Total</td>
<td>123</td>
<td>100.0</td>
</tr>
</tbody>
</table>

*a. Listwise deletion based on all variables in the procedure.*

### Table 8.0.1 Reliability Statistics

<table>
<thead>
<tr>
<th>Cronbach's Alpha</th>
<th>Cronbach's Alpha Based on Standardized Items</th>
<th>N of Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>.727</td>
<td>.789</td>
<td>6</td>
</tr>
</tbody>
</table>

### Table 8.0.2 Item Statistics

<table>
<thead>
<tr>
<th>q1</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4.7480</td>
<td>1.52892</td>
<td>123</td>
</tr>
<tr>
<td>q2</td>
<td>6.1301</td>
<td>4.07905</td>
<td>123</td>
</tr>
<tr>
<td>q3</td>
<td>14.3902</td>
<td>1.25374</td>
<td>123</td>
</tr>
<tr>
<td>q4</td>
<td>7.2846</td>
<td>7.02282</td>
<td>123</td>
</tr>
<tr>
<td>q5</td>
<td>12.1057</td>
<td>8.64492</td>
<td>123</td>
</tr>
<tr>
<td>q6</td>
<td>17.0325</td>
<td>11.15901</td>
<td>123</td>
</tr>
</tbody>
</table>

### Table 8.0.3 Inter-Item Correlation Matrix
<table>
<thead>
<tr>
<th></th>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
<th>q5</th>
<th>q6</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>1.000</td>
<td>.360</td>
<td>.197</td>
<td>.386</td>
<td>.271</td>
<td>.469</td>
</tr>
<tr>
<td>q2</td>
<td>.360</td>
<td>1.000</td>
<td>.192</td>
<td>.519</td>
<td>.471</td>
<td>.491</td>
</tr>
<tr>
<td>q3</td>
<td>.197</td>
<td>.192</td>
<td>1.000</td>
<td>.248</td>
<td>.223</td>
<td>.397</td>
</tr>
<tr>
<td>q4</td>
<td>.386</td>
<td>.519</td>
<td>.248</td>
<td>1.000</td>
<td>.411</td>
<td>.546</td>
</tr>
<tr>
<td>q5</td>
<td>.271</td>
<td>.471</td>
<td>.223</td>
<td>.411</td>
<td>1.000</td>
<td>.578</td>
</tr>
<tr>
<td>q6</td>
<td>.469</td>
<td>.491</td>
<td>.397</td>
<td>.546</td>
<td>.578</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 8.0.4 Item-Total Statistics

<table>
<thead>
<tr>
<th>Item</th>
<th>Scale Mean if Item Deleted</th>
<th>Scale Variance if Item Deleted</th>
<th>Corrected Total Correlation</th>
<th>Squared Multiple Correlation</th>
<th>Cronbach's Alpha if Item Deleted</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>56.9431</td>
<td>643.771</td>
<td>.473</td>
<td>.257</td>
<td>.732</td>
</tr>
<tr>
<td>q2</td>
<td>55.5610</td>
<td>551.998</td>
<td>.596</td>
<td>.378</td>
<td>.678</td>
</tr>
<tr>
<td>q3</td>
<td>47.3008</td>
<td>658.212</td>
<td>.358</td>
<td>.159</td>
<td>.742</td>
</tr>
<tr>
<td>q4</td>
<td>54.4065</td>
<td>455.633</td>
<td>.593</td>
<td>.395</td>
<td>.647</td>
</tr>
<tr>
<td>q5</td>
<td>49.5854</td>
<td>401.446</td>
<td>.596</td>
<td>.384</td>
<td>.645</td>
</tr>
<tr>
<td>q6</td>
<td>44.6585</td>
<td>289.337</td>
<td>.708</td>
<td>.547</td>
<td>.625</td>
</tr>
</tbody>
</table>

Table 8.0.5 Scale Statistics

<table>
<thead>
<tr>
<th>Mean</th>
<th>Variance</th>
<th>Std. Deviation</th>
<th>N of Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>61.6911</td>
<td>682.818</td>
<td>26.13078</td>
<td>6</td>
</tr>
</tbody>
</table>
8.1.1 Interrater reliability

To categorise students to the two different groups - liminal and post-liminal group - two raters were employed. The first rater was myself, and the second was an external computer science researcher to whom I had clearly explained the criteria on which his decision should be based.

Students who were still in the multi-structural level in functions were assigned to the liminal group. In contrast, those who demonstrated evidence of being in the relational level – a level where “a transformative quality of acquiring a threshold concept is evident” (Hamm, 2016:56) – were allocated to the post-liminal group. Particularly, students that did not solve correctly either task 5 or 6 (relational level) were categorised to the liminal group, whereas students that solved both tasks correctly were assigned to the post-liminal group. It is important to mention here that students in the post-liminal group answered correctly most of the questions. Some of these students made some minor mistakes on the second task but these mistakes did not refer to their ability to link together concepts and understand how these work together to solve the task (e.g., some errors occurred in lines 1,4,7, see page 296).

To evaluate the reliability of the scoring system, the level of agreement between the first rater and the second was calculated. Each rater categorised 123 students based on the criteria listed in the methodology chapter. Therefore, the inter-rater reliability of Cohen’s kappa was calculated, and it is depicted in the following tables. As can be seen from the symmetric measures table (Table 8.0.8), the k value indicates almost perfect agreement and with 95% Confidence interval of 0.778 to 0.963.
A discussion between the raters resolved any problematic cases which referred to cases that the two raters had categorised differently (seven students in total). Specifically, the two raters explored in detail the seven cases that they disagreed upon by examining the responses on the programming tasks in detail. After this discussion, the raters formed an agreement about whether the students were in liminal or in the post-liminal group.
The total sample of the students was split into two major categories: the liminal group and the post-liminal group. Each of these groups was further split according to the order they answered the questionnaire and the test. Specifically, the students that answered the questionnaire first formed the non-reversed group while students who solved the programming tasks first formed the reversed group. The following figure depicts the sample and the way it was split to test the research hypotheses. The numbers in the brackets indicate the participants in each group.

![Sample and the way it was split into subgroups](image)

8.1.2 Normality of the data

To determine if the data are normally distributed or not and, thus, decide whether parametric or non-parametric tests should be employed, the normality of the data was checked with the appropriate statistical tests. To this end, when the sample was less than 50, the Shapiro-Wilk
test was used and, in the case, that the sample was greater or equal than 50, the Kolmogorov-Smirnov was employed (Razali & Wah, 2011).

The results are presented for all the variables in this research. Specifically, Table 8.1.1 presents the normality test when the sample is split into two groups - liminal and post-liminal group -, table 8.1.2 presents the normality test when the sample is further split into the reversed and non-reversed group. The variables in the tables correspond to the affective constructs explored in the research: efficacyProg, refers to students’ efficacy beliefs in programming; taskValue, refers to the value the students allocate to the computer programming course; sense of belonging, refers to students’ sense of belonging in the computer science class; motivation, refers to students’ motivation and particularly intrinsic motivation in the computer science class; identity, refers to students’ computer science identity; self-evaluation, refers to students’ self-judgements on their performance in the programming test; bias and accuracy, refer to the calibration bias and calibration accuracy (metrics for students’ calibration, the literature chapter also mentions how these are calculated). The “liminal” column indicates if the group is in liminal (“yes”) or non-liminal space (“no”). The “reversed” column indicates if the group did the affective questionnaire first and then the programming test (“no”) or the other way around (“yes”).

Table 8.1.1 Liminal and post-liminal group

<table>
<thead>
<tr>
<th>Tests of Normality</th>
<th>Kolmogorov-Smirnov&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Shapiro-Wilk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Statistic</td>
<td>df</td>
</tr>
<tr>
<td>Liminal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>efficacyProg</td>
<td>.112</td>
</tr>
<tr>
<td></td>
<td>taskValue</td>
<td>.178</td>
</tr>
<tr>
<td></td>
<td>senseBelonging</td>
<td>motivation</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td>.172</td>
<td>.147</td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>.003</td>
<td>.020</td>
</tr>
<tr>
<td></td>
<td>.916</td>
<td>.900</td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>.004</td>
<td>.001</td>
</tr>
</tbody>
</table>

|                          | .916           | .916       | .969     | .866            | .837 | .837     |

|                          | .981           | .954       | .969     | .963            | .963 | .963     |

|                          | .962           | .943       | .969     | .966            | .963 | .963     |

|                          | .645           | .256       | .005     | .005            |      |          |

|                          | .200          | .200’      | .200’    | .200’           |      |          |

* This is a lower bound of the true significance.

a. Lilliefors Significance Correction

---

<table>
<thead>
<tr>
<th></th>
<th>No</th>
<th>efficacyProg</th>
<th>.100</th>
<th>18</th>
<th>.200’</th>
<th>.962</th>
<th>18</th>
<th>.645</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>taskValue</td>
<td>.113</td>
<td>18</td>
<td>.200’</td>
<td>.937</td>
<td>18</td>
<td>.256</td>
</tr>
<tr>
<td></td>
<td></td>
<td>senseBelonging</td>
<td>.229</td>
<td>18</td>
<td>.013</td>
<td>.837</td>
<td>18</td>
<td>.005</td>
</tr>
</tbody>
</table>

<p>| Tests of Normality       | | Kolmogorov-Smirnov | | Shapiro-Wilk |
|--------------------------| | Statistic | df | Sig. | Statistic | df | Sig. |
| Liminal                  | |            |    |     |            |    |     |
| Reversed                 | |            |    |     |            |    |     |
|                          |   | efficacyProg | .100 | 18 | .200’ | .962 | 18 | .645 |
|                          |   | taskValue    | .113 | 18 | .200’ | .937 | 18 | .256 |
|                          |   | senseBelonging | .229 | 18 | .013  | .837 | 18 | .005 |</p>
<table>
<thead>
<tr>
<th></th>
<th>motivation</th>
<th>identity</th>
<th>self-evaluation</th>
<th>Bias</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>.255</td>
<td>.240</td>
<td>.217</td>
<td>.195</td>
<td>.278</td>
</tr>
<tr>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>.053</td>
<td>.039</td>
<td>.200'</td>
<td>.245</td>
<td>.268</td>
</tr>
<tr>
<td>yes</td>
<td>.131</td>
<td>.003</td>
<td>.002</td>
<td>.248</td>
<td>.002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>efficacyProg</th>
<th>taskValue</th>
<th>senseBelonging</th>
<th>motivation</th>
<th>identity</th>
<th>self-evaluation</th>
<th>Bias</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>.129</td>
<td>.217</td>
<td>.206</td>
<td>.192</td>
<td>.118</td>
<td>.178</td>
<td>.159</td>
<td>.213</td>
</tr>
<tr>
<td>yes</td>
<td>.189</td>
<td>.182</td>
<td>.247</td>
<td>.182</td>
<td>.126</td>
<td>.228</td>
<td>.245</td>
<td>.268</td>
</tr>
<tr>
<td>yes</td>
<td>.053</td>
<td>.013</td>
<td>.200'</td>
<td>.053</td>
<td>.039</td>
<td>.200'</td>
<td>.053</td>
<td>.039</td>
</tr>
<tr>
<td>yes</td>
<td>.131</td>
<td>.003</td>
<td>.002</td>
<td>.131</td>
<td>.004</td>
<td>.004</td>
<td>.131</td>
<td>.004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>efficacyProg</th>
<th>taskValue</th>
<th>senseBelonging</th>
<th>motivation</th>
<th>identity</th>
<th>self-evaluation</th>
<th>Bias</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>.129</td>
<td>.217</td>
<td>.206</td>
<td>.192</td>
<td>.118</td>
<td>.178</td>
<td>.159</td>
<td>.213</td>
</tr>
<tr>
<td>yes</td>
<td>.189</td>
<td>.182</td>
<td>.247</td>
<td>.182</td>
<td>.126</td>
<td>.228</td>
<td>.245</td>
<td>.268</td>
</tr>
<tr>
<td>yes</td>
<td>.053</td>
<td>.013</td>
<td>.200'</td>
<td>.053</td>
<td>.039</td>
<td>.200'</td>
<td>.053</td>
<td>.039</td>
</tr>
<tr>
<td>yes</td>
<td>.131</td>
<td>.003</td>
<td>.002</td>
<td>.131</td>
<td>.004</td>
<td>.004</td>
<td>.131</td>
<td>.004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>efficacyProg</th>
<th>taskValue</th>
<th>senseBelonging</th>
<th>motivation</th>
<th>identity</th>
<th>self-evaluation</th>
<th>Bias</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>.129</td>
<td>.217</td>
<td>.206</td>
<td>.192</td>
<td>.118</td>
<td>.178</td>
<td>.159</td>
<td>.213</td>
</tr>
<tr>
<td>yes</td>
<td>.189</td>
<td>.182</td>
<td>.247</td>
<td>.182</td>
<td>.126</td>
<td>.228</td>
<td>.245</td>
<td>.268</td>
</tr>
<tr>
<td>yes</td>
<td>.053</td>
<td>.013</td>
<td>.200'</td>
<td>.053</td>
<td>.039</td>
<td>.200'</td>
<td>.053</td>
<td>.039</td>
</tr>
<tr>
<td>yes</td>
<td>.131</td>
<td>.003</td>
<td>.002</td>
<td>.131</td>
<td>.004</td>
<td>.004</td>
<td>.131</td>
<td>.004</td>
</tr>
</tbody>
</table>
The results indicate that there are inconsistencies in the distribution of some of the data. When that is the case, it is indicated in bold in the above tables, and it is also highlighted in the following paragraphs when a decision needs to be made on the tests to be employed.

8.1.3 Descriptive statistics

The following tables present the descriptive statistics for the following variables: self-efficacy, task value, sense of belonging, motivation and identity. I refer to these tables along the whole chapter to indicate the mean and median of the group that it is compared each time.

<table>
<thead>
<tr>
<th>Statistics</th>
<th>efficacyProg</th>
<th>taskValue</th>
<th>senseBelonging</th>
<th>motivation</th>
<th>identity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liminal</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>No</td>
<td>N</td>
<td>Valid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*a. Lilliefors Significance Correction*

*. This is a lower bound of the true significance.*
<table>
<thead>
<tr>
<th>Statistics</th>
<th>Liminal</th>
<th>Reverse</th>
<th>efficacy</th>
<th>taskValue</th>
<th>senseBelonging</th>
<th>motivation</th>
<th>identity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>6.01750</td>
<td>5.99611</td>
<td>5.45833</td>
<td>5.13889</td>
<td>4.95556</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td>5.93750</td>
<td>6.00000</td>
<td>5.62500</td>
<td>5.00000</td>
<td>5.30000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mode</td>
<td>5.750</td>
<td>5.600</td>
<td>6.000</td>
<td>6.000</td>
<td>5.600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Std. Deviation</td>
<td>.566992</td>
<td>.775304</td>
<td>.544018</td>
<td>.777544</td>
<td>.991862</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.1.3.2 Descriptive statistics for the liminal and post-liminal and reversed and non-reversed group

- Yes
  - N | Valid | Missing |
  - 80 | 80 | 80 | 80 | 80

- Mean: 4.59219, 5.50863, 4.50668, 4.58063, 3.91413
- Median: 4.75000, 5.80000, 4.75000, 4.62500, 4.00000
- Mode: 3.500, 6.000, 6.000, 5.000, 4.000
- Std. Deviation: 1.234026, 1.057922, 1.282791, 1.000553, 1.177151
- Minimum: 1.750, 2.600, 1.000, 2.500, 1.000
- Maximum: 7.000, 7.000, 6.000, 6.000, 6.000

a. Multiple modes exist. The smallest value is shown.
<table>
<thead>
<tr>
<th>Yes</th>
<th>N</th>
<th>Valid</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>N</td>
<td>Valid</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>4.800</td>
<td>7.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>4.500</td>
<td>6.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>5.36680</td>
<td>5.73080</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>5.25000</td>
<td>6.00000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mode</td>
<td>6.000</td>
<td>6.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Std. Deviation</td>
<td>0.862369</td>
<td>0.901147</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Minimum</td>
<td>3.625</td>
<td>4.160</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>7.000</td>
<td>7.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>5.11023</td>
<td>5.74591</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>5.25000</td>
<td>5.91500</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mode</td>
<td>5.000*</td>
<td>6.500</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Std. Deviation</td>
<td>1.150974</td>
<td>0.776909</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Minimum</td>
<td>2.500</td>
<td>4.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum</td>
<td>6.500</td>
<td>7.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>4.39569</td>
<td>5.41862</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>4.43500</td>
<td>5.70000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mode</td>
<td>3.500*</td>
<td>6.000*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Std. Deviation</td>
<td>1.216406</td>
<td>1.139659</td>
<td></td>
</tr>
</tbody>
</table>

| yes | N | Valid | 22 | 22 | 22 | 22 | 22 |
|     | Min | 3.625 | 4.160 |
|     | Max | 7.000 | 7.000 |
|     | Mean | 5.11023 | 5.74591 |
|     | Median | 5.25000 | 5.91500 |
|     | Mode | 5.000* | 6.500 |
|     | Std. Deviation | 1.150974 | 0.776909 |

| yes | N | Valid | 58 | 58 | 58 | 58 | 58 |
|     | Min | 2.500 | 4.000 |
|     | Max | 6.500 | 7.000 |
|     | Mean | 4.39569 | 5.41862 |
|     | Median | 4.43500 | 5.70000 |
|     | Mode | 3.500* | 6.000* |
|     | Std. Deviation | 1.216406 | 1.139659 |
Minimum | 1.750 | 2.600 | 1.000 | 2.500 | 1.000
Maximum | 7.000 | 7.000 | 6.000 | 6.000 | 6.000

a. Multiple modes exist. The smallest value is shown

### 8.2 Calibration in liminal and post-liminal space

One part of the first research question of this phase of the study is concerned with students’ calibration ability and if their estimations of their performance are consistent with their actual performance when students are in liminal and post-liminal spaces. Specifically, the question of interest is the following:

- Does being in liminal or post-liminal space affect students’ calibration, self-efficacy, task value, sense of belonging, motivation and computer science identity?

To this end, the calibration bias and calibration accuracy were calculated for each student as described in the literature section (2.4.3.1). The following subsection presents the results and the comparison between the two groups - the liminal and post-liminal group.

#### 8.2.1 Descriptive statistics

The following table presents the descriptive statistics of each group. These were not included in the previous tables as all the four variables depicted in the following table refer to a separate construct that of calibration. As it can be seen from Table 8.2.1, the mean score of students’ performances in liminal space is 45.8 with a median of 46.7 and for students in the post-liminal group is 91.08 with a median of 90.0. Regarding the self-evaluation variable, the mean score for the liminal group is 53.25, with a median of 50.0 and for the post-liminal group is 79.06 with a median of 80.0.
Table 8.2.1. Descriptive statistics of the calibration variables

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Liminal performance</th>
<th>self-evaluation</th>
<th>bias</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>No N Valid</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>Missing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mean</td>
<td>91.0814</td>
<td>79.06</td>
<td>-1.19</td>
<td>8.44</td>
</tr>
<tr>
<td>Median</td>
<td>90.0000</td>
<td>80.00</td>
<td>-1.00</td>
<td>9.00</td>
</tr>
<tr>
<td>Mode</td>
<td>88.00</td>
<td>80</td>
<td>-1</td>
<td>9</td>
</tr>
<tr>
<td>Std. Deviation</td>
<td>5.65783</td>
<td>1.716</td>
<td>1.708</td>
<td>1.368</td>
</tr>
<tr>
<td>Yes N Valid</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Missing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mean</td>
<td>45.8938</td>
<td>53.25</td>
<td>.63</td>
<td>8.63</td>
</tr>
<tr>
<td>Median</td>
<td>46.7500</td>
<td>50.0</td>
<td>.50</td>
<td>9.00</td>
</tr>
<tr>
<td>Mode</td>
<td>29.50(^a)</td>
<td>50</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Std. Deviation</td>
<td>17.73641</td>
<td>20.17</td>
<td>1.760</td>
<td>1.321</td>
</tr>
</tbody>
</table>

a. Multiple modes exist. The smallest value is shown

8.2.2 Correlations

The following tables present the correlations between self-evaluation and the rest of the examined affective dimension variables (table 8.2.2.1) as well as the correlation between students’ performance and calibration bias and accuracy (8.2.2.2). Because the examined variables were not normally distributed, Spearman’s rho correlation was used.
<table>
<thead>
<tr>
<th>No</th>
<th>Self-evaluation Coefficient</th>
<th>Liminal</th>
<th></th>
<th>efficacy</th>
<th>task</th>
<th>sense</th>
<th>motivation</th>
<th>identity</th>
<th>performance</th>
<th>Self-evaluation bias</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Self-evaluation Coefficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Spearman's rho</th>
<th>N</th>
<th>Self-evaluation Correlation Coefficient</th>
<th>Liminal</th>
<th></th>
<th>efficacy</th>
<th>task</th>
<th>sense</th>
<th>motivation</th>
<th>identity</th>
<th>performance</th>
<th>Self-evaluation bias</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Self-evaluation Coefficient</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*: Correlation is significant at the 0.05 level (2-tailed).

**: Correlation is significant at the 0.01 level (2-tailed).
Table 8.2.2.2 Correlation table between performance and calibration bias and accuracy

<table>
<thead>
<tr>
<th>Correlations</th>
<th>Liminal</th>
<th>performance</th>
<th>bias</th>
<th>newaccuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spearman's rho</td>
<td>No</td>
<td>Correlation Coefficient</td>
<td>1.000</td>
<td>-.053</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sig. (2-tailed)</td>
<td>.</td>
<td>.735</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>bias</td>
<td>Correlation Coefficient</td>
<td>-.053</td>
<td>1.000</td>
<td>.837**</td>
</tr>
<tr>
<td></td>
<td>Sig. (2-tailed)</td>
<td>.735</td>
<td>.</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>newaccuracy</td>
<td>Correlation Coefficient</td>
<td>-.065</td>
<td>.837**</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td>Sig. (2-tailed)</td>
<td>.680</td>
<td>.000</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>Yes</td>
<td>Correlation Coefficient</td>
<td>1.000</td>
<td>-.341**</td>
<td>.245*</td>
</tr>
<tr>
<td></td>
<td>Sig. (2-tailed)</td>
<td>.</td>
<td>.002</td>
<td>.029</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>bias</td>
<td>Correlation Coefficient</td>
<td>-.341**</td>
<td>1.000</td>
<td>-.469**</td>
</tr>
<tr>
<td></td>
<td>Sig. (2-tailed)</td>
<td>.002</td>
<td>.</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>newaccuracy</td>
<td>Correlation Coefficient</td>
<td>.245*</td>
<td>-.469**</td>
<td>1.000</td>
</tr>
<tr>
<td></td>
<td>Sig. (2-tailed)</td>
<td>.029</td>
<td>.000</td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).
What it is interesting from the above tables is that students’ self-evaluation beliefs when they are in liminal space correlate with all the other affective constructs while in the other group students’ self-evaluation beliefs correlate only with their efficacy in programming and performance.

8.2.3 General comparison between groups

To examine students’ calibration ability, the calibration bias and calibration accuracy for each student was calculated for both groups, and the results for each group were presented in table 8.2.1. The students in the liminal space have a calibration bias mean score of .63, whereas the students in the post-liminal space have a negative mean score of -1.19. This suggests that students in the post-liminal space tend to underestimate themselves in comparison with students in liminal space. Both groups have good calibration accuracy: 8.63 and 8.44.

To examine if these differences are statistically significant, the following hypotheses were tested:

H0a: There is no difference in the calibration bias score between the liminal and the post-liminal group.

H0b: There is no difference in the calibration accuracy score between the liminal and the post-liminal group.

H1a: There is a difference in the calibration bias score between the liminal and the post-liminal group.
H1b: There is a difference in the calibration accuracy score between the liminal and the post-liminal group.

Table 8.2.3.1 Ranks for the calibration variables

<table>
<thead>
<tr>
<th>Ranks</th>
<th>Liminal</th>
<th>N</th>
<th>Mean Rank</th>
<th>Sum of Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias</td>
<td>No</td>
<td>43</td>
<td>40.59</td>
<td>1745.50</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>80</td>
<td>73.51</td>
<td>5880.50</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>accuracy</td>
<td>No</td>
<td>43</td>
<td>59.29</td>
<td>2549.50</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>80</td>
<td>63.46</td>
<td>5076.50</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>123</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.2.3.2 Mann-Whitney test for the calibration variables

<table>
<thead>
<tr>
<th>Test Statisticsa</th>
<th>bias</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mann-Whitney U</td>
<td>799.50</td>
<td>1603.500</td>
</tr>
<tr>
<td>Wilcoxon W</td>
<td>1745.50</td>
<td>2549.500</td>
</tr>
<tr>
<td>Z</td>
<td>-4.956</td>
<td>-.642</td>
</tr>
<tr>
<td>Asymp. Sig. (2-tailed)</td>
<td>.000</td>
<td>.521</td>
</tr>
</tbody>
</table>

a. Grouping Variable: Liminal

Since the data were not normally distributed (Table 8.1.1), the Mann and Whitney test was employed. The results depicted in the above tables indicate a statistically significant difference in the calibration bias between students in the liminal and students in the post-
liminal group \((z=-4.956, p<.05, d=0.446)\) and, thus, the null hypothesis \(H_0a\) is rejected, and the \(H_1a\) is accepted.

8.3 **Self-efficacy in liminal and post-liminal space**

Another part of the 5\(^{th}\) and 6\(^{th}\) research questions of the study concerns with students’ self-efficacy when they are in a liminal and post-liminal space. Specifically, the questions of interest are the following:

a. Does being in liminal or post-liminal space affect students’ calibration, self-efficacy, task value, sense of belonging, motivation and computer science identity?

b. Does students’ encounter with troublesome knowledge impact their self-efficacy, task value, sense of belonging, motivation and computer science identity?

8.3.1 **Comparison between liminal and post-liminal group**

Table 8.1.3.1 indicates that students in the post-liminal space demonstrate higher levels of self-efficacy in programming than students who are in liminal space. Specifically, the mean score for the self-efficacy in programming is 5.6 for the post-liminal group and 4.5 for students in the liminal group. This suggests that students in liminal space have average levels of self-efficacy in programming.

To examine if this difference between students in liminal and students in post-liminal space is statistically significant, the independent sample t-test was used since the specific data for both groups were normally distributed (Table 8.1.1). The null hypothesis and the alternative hypothesis are stated below:
H01: There is no difference in the self-efficacy score between students in liminal and students in post-liminal space

H11: There is a difference in the self-efficacy score between students in liminal and students in post-liminal space

The results indicate that the difference in the self-efficacy score is statistically significant ($t=0.006$, $p<.00$, $r=0.44$) with good effect size, suggesting that students in the liminal space demonstrate significantly lower levels of self-efficacy than do students in the post-liminal space.

| Group Statistics
<table>
<thead>
<tr>
<th>Liminal</th>
<th>N</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Std. Error Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>efficacyProg</td>
<td>No</td>
<td>43</td>
<td>5.63919</td>
<td>.812763</td>
</tr>
<tr>
<td>Yes</td>
<td>80</td>
<td>4.59219</td>
<td>1.234026</td>
<td>.137968</td>
</tr>
</tbody>
</table>

Table 8.3.1.1 Descriptive statistics – self-efficacy variable

| Independent Samples Test
<table>
<thead>
<tr>
<th>Levene's Test for Equality of Variances</th>
<th>t-test for Equality of Means</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Sig.</td>
</tr>
<tr>
<td>efficacyProg</td>
<td>Equal variances assumed</td>
</tr>
</tbody>
</table>

Table 8.3.1.2 Independent Samples T-Test – self-efficacy
8.3.2 Comparison within liminality groups and between reversed and non-reversed group

To test if and how students’ self-efficacy is influenced when students encounter troublesome knowledge, the sample was further split depending on whether the students did the test before or after the affective dimension questionnaire. The descriptive statistics for each group are presented in table 8.1.3.2.

As it can be seen from this table, there is a difference in the self-efficacy score for students that did the questionnaire after the test and students that did the questionnaire before and this difference is evident both in the post-liminal and the liminal group. To test if this difference is significant within the two groups the independent sample T-test was used for the post-liminal group, and the Mann and Whitney test was employed for the liminal group since the data were not normally distributed. The null and alternative hypotheses tested are listed below:

H02a: There is no difference in the self-efficacy scores of students in liminal space that did the questionnaire before the test (non-reversed) and students in liminal space that did the questionnaire after the test (reversed).

H02b: There is no difference in the self-efficacy scores of students in post-liminal space that did the questionnaire before the test and students in post-liminal space that did the questionnaire after the test.
H12a: There is a difference in the self-efficacy scores of students in liminal space that did the questionnaire before the test and students in liminal space that did the questionnaire after the test.

H12b: There is a difference in the self-efficacy scores of students in post-liminal space that did the questionnaire before the test and students in post-liminal that did the questionnaire after the test.

Table 8.3.2.1 Descriptive statistics – self-efficacy in post-liminal group and in reverse and non-reverse condition

<table>
<thead>
<tr>
<th>Group Statistics</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Liminal</td>
<td>reverse</td>
<td>N</td>
<td>Mean</td>
<td>Std. Deviation</td>
<td>Std. Error Mean</td>
</tr>
<tr>
<td>No efficiency</td>
<td>prog normal</td>
<td>18</td>
<td>6.01750</td>
<td>.566992</td>
<td>.133641</td>
</tr>
<tr>
<td></td>
<td>not</td>
<td>25</td>
<td>5.36680</td>
<td>.862369</td>
<td>.172474</td>
</tr>
</tbody>
</table>

Table 8.3.2.2 Independent Samples Test - self-efficacy comparison in post-liminal and in reverse and non-reverse condition

<table>
<thead>
<tr>
<th>Independent Samples Test</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Liminal</td>
<td>Levene's Test for Equality of Variances</td>
<td>t-test for Equality of Means</td>
<td>F</td>
<td>Sig.</td>
<td>t</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>F</td>
<td>Sig.</td>
<td>t</td>
</tr>
</tbody>
</table>

254
Table 8.3.2.3 Rank - self-efficacy in liminal group and in reverse and non-reverse condition

<table>
<thead>
<tr>
<th>Rank</th>
<th>Liminal</th>
<th>reverse</th>
<th>N</th>
<th>Mean Rank</th>
<th>Sum of Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>efficacyProg</td>
<td>normal</td>
<td>22</td>
<td>51.00</td>
<td>1122.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not</td>
<td>58</td>
<td>36.52</td>
<td>2118.00</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.3.2.4 Mann-Whitney test for self-efficacy in the liminal group and in reverse and non-reverse condition

<table>
<thead>
<tr>
<th>Test Statisticsa</th>
<th>Liminal</th>
<th>efficacyProg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Mann-Whitney U</td>
<td>.407.000</td>
</tr>
<tr>
<td></td>
<td>Wilcoxon W</td>
<td>2118.000</td>
</tr>
<tr>
<td></td>
<td>Z</td>
<td>-.2.491</td>
</tr>
<tr>
<td></td>
<td>Asymp. Sig. (2-tailed)</td>
<td>.013</td>
</tr>
</tbody>
</table>

a. Grouping Variable: reverse

The results suggest that the differences in students’ self-efficacy levels are statistically significant both for the liminal (z=-2.491, p<.05, r=0.27) and the post-liminal group (t=2.792, p<.05, r=0.40). This indicates that students’ self-efficacy levels are negatively
influenced when they encounter troublesome knowledge, and this is valid both for students in the liminal and students in the post-liminal group.

8.4 Task value in liminal and post-liminal space

Another part of the 5th and 6th research questions was about students’ task value when they are in liminal and post-liminal space. Specifically, the questions of interest were the following:

a. Does being in liminal or post-liminal space affect students’ calibration, self-efficacy, task value, sense of belonging, motivation and computer science identity?

b. Does students’ encounter with troublesome knowledge impact their self-efficacy, task value, sense of belonging, motivation and computer science identity?

8.4.1 Comparison between liminal and post-liminal group

As it can be seen from the table 8.1.3.1, the mean score for students in liminal and students in post-liminal space does not differ much, with the former reaching a mean score of 5.84 and the latter reaching a score of 5.50. Additionally, this difference is not statistically significant as indicated by the Mann-Whitney test (Table 8.4.1.1) suggesting that generally, students’ task value is not influenced by whether students are in liminal spaces or not. The results of the Mann-Whitney test are presented in the following tables (8.4.1.1 and 8.4.1.2) along with the null and alternative hypotheses that were tested.
H04: There is no difference in the task value scores between students in liminal and students in post-liminal space

H14: There is a difference in the task value scores between students in liminal and students in post-liminal space

Table 8.4.1.1 Ranks – Task value in the liminal and post-liminal group

<table>
<thead>
<tr>
<th>Ranks</th>
<th>Liminal</th>
<th>N</th>
<th>Mean Rank</th>
<th>Sum of Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskValue</td>
<td>No</td>
<td>43</td>
<td>68.85</td>
<td>2960.50</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>80</td>
<td>58.32</td>
<td>4665.50</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>123</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.4.1.2 Mann-Whiney test - task value in the liminal and post-liminal group

<table>
<thead>
<tr>
<th>Test Statisticsa</th>
<th>taskValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mann-Whitney U</td>
<td>1425.500</td>
</tr>
<tr>
<td>Wilcoxon W</td>
<td>4665.500</td>
</tr>
<tr>
<td>Z</td>
<td>-1.566</td>
</tr>
<tr>
<td>Asymp. Sig. (2-tailed)</td>
<td>.117</td>
</tr>
</tbody>
</table>

a. Grouping Variable: Liminal
8.4.2 Comparison within liminality groups and between reversed and non-reversed group

To test if students’ scores are influenced by whether students encountered with troublesome knowledge or not, these groups were further split into students that did the test before and after the questionnaire. The null and alternative hypotheses that were tested are listed below:

H05a: There is no difference in the task value scores of students in liminal space that did the questionnaire before the test (non-reversed) and students in liminal space that did the questionnaire after the test (reversed).

H05b: There is no difference in the task value scores of students in post-liminal space that did the questionnaire before the test and students in post-liminal space that did the questionnaire after the test.

H15a: There is a difference in the task value scores of students in liminal space that did the questionnaire before the test and students in liminal space that did the questionnaire after the test.

H15b: There is a difference in the task value scores of students in post-liminal space that did the questionnaire before the test and students in post-liminal space that did the questionnaire after the test.

The results of the Mann-Whitney test employed (Tables 8.4.2.1 and 8.4.2.2) suggest that students’ task value scores were not influenced by the troublesome knowledge they encountered, on the contrary to what was evident with their self-efficacy. Therefore, the null hypotheses could not be rejected.
Table 8.4.2.1 Ranks – Task value for the liminal and post-liminal group in reverse and not reverse condition

<table>
<thead>
<tr>
<th></th>
<th>reverse</th>
<th>N</th>
<th>Mean Rank</th>
<th>Sum of Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liminal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No taskValue</td>
<td>normal</td>
<td>18</td>
<td>23.75</td>
<td>427.50</td>
</tr>
<tr>
<td></td>
<td>not</td>
<td>25</td>
<td>20.74</td>
<td>518.50</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes taskValue</td>
<td>normal</td>
<td>22</td>
<td>44.14</td>
<td>971.00</td>
</tr>
<tr>
<td></td>
<td>not</td>
<td>58</td>
<td>39.12</td>
<td>2269.00</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.4.2.2 Mann-Whitney test - Task value for the liminal and post-liminal group in reverse and not reverse condition

|               |         | taskValue |           |           |
|---------------|---------|-----------|-----------|
| Liminal       |         |           |           |           |
| No            | Mann-Whitney U | 193.500  |           |           |
|               | Wilcoxon W    | 518.500  |           |           |
|               | Z             | -.779     |           |           |
|               | Asymp. Sig. (2-tailed) | .436     |           |           |
| Yes           | Mann-Whitney U | 558.000  |           |           |
|               | Wilcoxon W    | 2269.000 |           |           |
|               | Z             | -.864     |           |           |
|               | Asymp. Sig. (2-tailed) | .388     |           |           |

a. Grouping Variable: reverse
8.5 Sense of belonging, motivation and identity in liminal and post-liminal space

The final part of the 5th and 6th research questions was about students’ sense of belonging, motivation and identity when they are in liminal and post-liminal space. Specifically, the questions of interest were the following:

a. Does being in liminal or post-liminal space affect students’ calibration, self-efficacy, task value, sense of belonging, motivation and computer science identity?

b. Does students’ encounter with troublesome knowledge impact their self-efficacy, task value, sense of belonging, motivation and computer science identity?

8.5.1 Comparison between liminal and post-liminal group

Table 8.1.3.1 indicates that students in liminal space scored less in the variables of sense of belonging, motivation and identity than did students in post-liminal space. Particularly, the students in post-liminal space demonstrate a mean score of 4.95, 4.81, and 4.60 in these variables, whereas the students in liminal space have a mean score of 4.50, 4.58 and 3.91 correspondingly. To test if these differences between the two groups are statistically significant, the Mann-Whitney test was employed since the data were not normally distributed. The null and alternative hypotheses that were tested are listed below:

H07: There is no difference in the sense of belonging score between students in liminal and students in post-liminal space
H17: There is a difference in the sense of belonging score between students in liminal and students in post-liminal space

H08: There is no difference in the motivation score between students in liminal and students in post-liminal space

H18: There is a difference in the motivation score between students in liminal and students in post-liminal space

H09: There is no difference in the identity score between students in liminal and students in post-liminal space

H19: There is a difference in the identity score between students in liminal and students in post-liminal space

The results (Tables 8.5.1.1, and 8.5.1.2) suggest that the difference in the scores between these two groups is statistically significant only for the identity variable and in favour of the post-liminal group ($z=-2.967$, $p<.05$, $r=.26$). This suggests that students in liminal space demonstrate significantly lower levels of computer science identity than the students in the post-liminal space.

Table 8.5.1.1 Ranks – sense of belonging, motivation, identity for the liminal and post-liminal group

<table>
<thead>
<tr>
<th>Ranks</th>
<th>Liminal</th>
<th>N</th>
<th>Mean Rank</th>
<th>Sum of Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>senseBelonging</td>
<td>No</td>
<td>43</td>
<td>68.98</td>
<td>2966.00</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>80</td>
<td>58.25</td>
<td>4660.00</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>123</td>
<td></td>
<td></td>
</tr>
<tr>
<td>motivation</td>
<td>No</td>
<td>43</td>
<td>68.27</td>
<td>2935.50</td>
</tr>
</tbody>
</table>
Table 8.5.1.2 Mann-Whitney test - sense of belonging, motivation, identity for the liminal and post-liminal group

<table>
<thead>
<tr>
<th></th>
<th>Yes</th>
<th>58.63</th>
<th>4690.50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>123</td>
<td></td>
</tr>
<tr>
<td>identity</td>
<td>No</td>
<td>74.97</td>
<td>3223.50</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>55.03</td>
<td>4402.50</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>123</td>
<td></td>
</tr>
</tbody>
</table>

Test Statistics

<table>
<thead>
<tr>
<th></th>
<th>senseBelonging</th>
<th>motivation</th>
<th>identity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mann-Whitney U</td>
<td>1420.000</td>
<td>1450.500</td>
<td>1162.500</td>
</tr>
<tr>
<td>Wilcoxon W</td>
<td>4660.000</td>
<td>4690.500</td>
<td>4402.500</td>
</tr>
<tr>
<td>Z</td>
<td>-1.604</td>
<td>-1.438</td>
<td>-2.967</td>
</tr>
<tr>
<td>Asymp. Sig. (2-tailed)</td>
<td>.109</td>
<td>.150</td>
<td>.003</td>
</tr>
</tbody>
</table>

a. Grouping Variable: Liminal

8.5.2 Comparison within liminality groups and between reversed and non-reversed group

To examine if these scores are influenced by troublesome knowledge - whether students were given the questionnaire before or after the test - the sample was further split into students that did the test after the questionnaire (non-reversed) and before (reversed). The statistical tests employed were the following:

- the Mann-Whitney test was employed to compare the sense of belonging, motivation and identity score for students in post-liminal space that did the test before and after the questionnaire
• the Mann-Whitney test was employed to compare the sense of belonging, and identity score for students in liminal space that did the test before and after the questionnaire

• the Independent sample t-test was employed to compare the motivation score for students in liminal space that did the test before and after the questionnaire

As the following table (8.5.2.0) depicts, for the post-liminal group, the mean score for the sense of belonging of students in the non-reversed group was 5.45 and in the reversed group was 4.59. For the liminal group, the corresponding scores were 5.23 and 4.22. For the motivation variable, the mean score for the post-liminal group and the non-reversed group was 5.13 and for the reversed group was 4.57 while for the liminal group was 5.13 and 4.36 correspondingly. Finally, for the identity variable, the mean score for the post-liminal group and for the non-reversed group was 4.95 and 4.34 for the reversed group, and for the liminal group was 4.29 and 3.77 correspondingly. Table 8.5.2.0 depicts these differences.

Table 8.5.2.0 Within-group differences depending on the order the test and the questionnaire was distributed

<table>
<thead>
<tr>
<th>Sense of belonging</th>
<th>Liminal group</th>
<th>Post-liminal group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reverse</td>
<td>4.22</td>
<td>5.23</td>
</tr>
<tr>
<td>Non-reverse</td>
<td>4.59</td>
<td>5.45</td>
</tr>
<tr>
<td>Motivation</td>
<td>4.36</td>
<td>5.13</td>
</tr>
<tr>
<td>CS identity</td>
<td>3.77</td>
<td>4.29</td>
</tr>
<tr>
<td>Reverse</td>
<td>4.34</td>
<td>4.95</td>
</tr>
<tr>
<td>Non-reverse</td>
<td>4.95</td>
<td></td>
</tr>
</tbody>
</table>

To test if the aforementioned differences were statistically significant, the following null and alternative hypotheses were tested:
H010a: There is no difference in the sense of belonging scores of students in liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

H010b: There is no difference in the sense of belonging scores of students in post-liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

H110a: There is a difference in the sense of belonging scores of students in liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

H110b: There is a difference in the sense of belonging scores of students in post-liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

H011a: There is no difference in the motivation scores of students in liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

H011b: There is no difference in the motivation scores of students in post-liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

H111a: There is a difference in the motivation scores of students in liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

H111b: There is a difference in the motivation scores of students in post-liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

H012a: There is no difference in the identity scores of students in liminal space that did the questionnaire before the test and students that did the questionnaire after the test.
H012b: There is no difference in the identity scores of students in post-liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

H112a: There is a difference in the identity scores of students in liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

H112b: There is a difference in the identity scores of students in post-liminal space that did the questionnaire before the test and students that did the questionnaire after the test.

The results (Tables 8.5.2.1, 8.5.2.2, 8.5.2.3, and 8.5.2.4) suggest that for the liminal group the differences are statistically significant for the sense of belonging ($z=-3.052$, $p<.05$, $r=0.341$), motivation ($t=3.238$, $p<.05$, $r=0.40$) but not the identity variable, while for the post-liminal group the results are significant only for the sense of belonging ($-3.253$, $p<.05$, $r=0.49$). These suggest that for students in liminal space, troublesome knowledge has an impact on their affective dimensions and particularly the sense of belonging and motivation while for the post-liminal group, troublesome knowledge affects the sense of belonging only.

<table>
<thead>
<tr>
<th>Ranks</th>
<th>Liminal</th>
<th>reverse</th>
<th>N</th>
<th>Mean Rank</th>
<th>Sum of Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>senseBelonging</td>
<td>normal</td>
<td>18</td>
<td>29.22</td>
<td>526.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not</td>
<td>25</td>
<td>16.80</td>
<td>420.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>motivation</td>
<td>normal</td>
<td>18</td>
<td>25.25</td>
<td>454.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not</td>
<td>25</td>
<td>19.66</td>
<td>491.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>identity</td>
<td>normal</td>
<td>18</td>
<td>26.22</td>
<td>472.00</td>
</tr>
</tbody>
</table>

Table 8.5.2.1 Ranks - sense of belonging, motivation, identity for the liminal and post-liminal group and the reverse and not reverse condition
<table>
<thead>
<tr>
<th></th>
<th>senseBelonging</th>
<th>motivation</th>
<th>identity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>normal</td>
<td>not</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>22</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td></td>
<td>53.30</td>
<td>35.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1172.50</td>
<td>2067.50</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>80</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>47.27</td>
<td>37.93</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1040.00</td>
<td>2200.00</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.5.2.2 Mann-Whitney test - sense of belonging, motivation, identity for the liminal and post-liminal group and the reverse and not reverse condition

<table>
<thead>
<tr>
<th>Test Statistics\a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liminal</td>
</tr>
<tr>
<td>No</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

a. Grouping Variable: reverse
Table 8.5.2.3 Descriptive statistics - Motivation in the liminal group and reverse and not reverse condition

<table>
<thead>
<tr>
<th>Group Statistics</th>
<th>reverse</th>
<th>N</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Std. Error Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes Motivation normal</td>
<td>22</td>
<td>5.13636</td>
<td>.675803</td>
<td>.144082</td>
<td></td>
</tr>
<tr>
<td>not</td>
<td>58</td>
<td>4.36983</td>
<td>1.027044</td>
<td>.134857</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.5.2.4 Independent Samples Test - Motivation in the liminal group and reverse and not reverse condition

<table>
<thead>
<tr>
<th>Independent Samples Test</th>
<th>Levene's Test for Equality of Variances</th>
<th>t-test for Equality of Means</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>Sig.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes motivation Equal variances assumed</td>
<td>5.346</td>
<td>.023</td>
</tr>
<tr>
<td>Equal variances not assumed</td>
<td>3.8</td>
<td>.000</td>
</tr>
<tr>
<td></td>
<td>84</td>
<td>6</td>
</tr>
</tbody>
</table>
8.6 Correlations with performance

This section presents the correlations depicted between the students’ performance and the examined affective variables. When the data were normally distributed, Pearson correlation was used, and it is indicated in the corresponding correlation coefficient with the symbol “P” while Spearman’s rho was employed when the data were not normally distributed.

Table 8.6.1 Correlations in liminal and no liminal group

<table>
<thead>
<tr>
<th>Liminal</th>
<th>Performance</th>
<th>Correlation Coefficient</th>
<th>N</th>
<th>Sig. (2-tailed)</th>
<th>Significance</th>
<th>N</th>
<th>Sig. (2-tailed)</th>
<th>Significance</th>
<th>N</th>
<th>Sig. (2-tailed)</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Performance</td>
<td>Correlation Coefficient</td>
<td>.011 (P)</td>
<td>.300</td>
<td>.195</td>
<td>.073</td>
<td>.036</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.944</td>
<td>.051</td>
<td>.211</td>
<td>.641</td>
<td>.821</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Performance</td>
<td>Correlation Coefficient</td>
<td>.382**</td>
<td>.435**</td>
<td>.339**</td>
<td>.258*</td>
<td>.297**</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.000</td>
<td>.000</td>
<td>.002</td>
<td>.021</td>
<td>.007</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**. Correlation is significant at the 0.01 level (2-tailed).
*. Correlation is significant at the 0.05 level (2-tailed).

As it can be seen from the table 8.6.1, the performance variable in the liminal group forms statistically significant correlations with all the affective variables, with the task value being the strongest one \( r^2 = .435 \). Interestingly, in the post-liminal group, the performance variable does not form any statistically significant correlations with the other variables.
Table 8.6.2 Correlations for reverse and non-reversed group

<table>
<thead>
<tr>
<th>Liminal</th>
<th>Reverse</th>
<th>Performance</th>
<th>Correlation Coefficient</th>
<th>efficacy.Prog</th>
<th>task.Value</th>
<th>senseBelonging</th>
<th>motivation</th>
<th>identity</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>Performance</td>
<td>Correlation Coefficient</td>
<td>-.127 (P)</td>
<td>.206 (P)</td>
<td>-.105</td>
<td>-.279</td>
<td>-.369</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sig. (2-tailed)</td>
<td>.617</td>
<td>.413</td>
<td>.678</td>
<td>.263</td>
<td>.131</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Performance</td>
<td>Correlation Coefficient</td>
<td>.076</td>
<td>.336</td>
<td>.414*</td>
<td>.396*</td>
<td>.306</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sig. (2-tailed)</td>
<td>.716</td>
<td>.100</td>
<td>.039</td>
<td>.050</td>
<td>.136</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>no</td>
<td>Performance</td>
<td>Correlation Coefficient</td>
<td>-.190 (P)</td>
<td>.092</td>
<td>.449*</td>
<td>.588*</td>
<td>.554*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sig. (2-tailed)</td>
<td>.397</td>
<td>.684</td>
<td>.036</td>
<td>.004</td>
<td>.007</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>Performance</td>
<td>Correlation Coefficient</td>
<td>.511**(P)</td>
<td>.529**</td>
<td>.294*(P)</td>
<td>.120(P)</td>
<td>.199</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sig. (2-tailed)</td>
<td>.000</td>
<td>.000</td>
<td>.025</td>
<td>.370</td>
<td>.134</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>58</td>
<td>58</td>
<td>58</td>
<td>58</td>
<td>58</td>
<td></td>
</tr>
</tbody>
</table>

**. Correlation is significant at the 0.01 level (2-tailed).
*. Correlation is significant at the 0.05 level (2-tailed).

Table 8.6.2 depicts the correlations between performance and the affective variables when the sample is split by the liminal group and by the way the students responded to the questionnaire and the test. What is noticeable from this table is that most of the significant correlations are once more depicted in the liminal group. Specifically, in the liminal group and the non-reversed group, the sense of belonging, the motivation and the identity variables form statistically significant correlations that are stronger than the correlations depicted in the reversed group. The only exception is that the self-efficacy in programming and task
value form statistically significant correlations with performance only in the reversed group. For the post-liminal group, there is evidence of statistically significant correlations between performance and the sense of belonging and motivation only for the reversed group. No other significant correlations are depicted in the post-liminal group.

8.7 Regression Model

The current section presents the prediction models of liminality. The research question that this section endeavours to answer is the following:

- Can a model be used that predicts if students are in liminal or a post-liminal state?

Specifically, this analysis aimed to predict the likelihood of liminal or post-liminal space occurring, based on the affective dimensions of learning examined in this thesis. To this end, two scenarios were considered: the first one, takes into consideration the students’ self-evaluations as well as the other affective constructs and the second one takes into consideration only the affective constructs and attempts to find a model that can predict students’ liminal state without administering a test. For this reason, the second scenario only considers the group that did the affective questionnaire first. In both cases, the number of predictors was based on the rule of thumb that the model should have at least ten times as many participants as predictors (Brace et al., 2016). In the first scenario, two models were created, and both predict the liminality state for the reversed and non-reversed group. In the second scenario, one model was created, which predicts the liminal state of the non-reversed group.
8.7.1 Assumptions

To explore the predictor variables that can be used to predict the likelihood of the liminality (categorical) outcome occurring, the binary logistic regression was employed. The assumptions for the logistic regression that were tested are the following:

a. the outcome variable must be categorical with two possible outcomes which must be coded as 0 and 1,

b. the predictors can be numerical and categorical,

c. there must be reasonably linearity between the predictors and the outcome. To assess linearity, new variables that represent the interaction between the predictor variables and the natural log of that variables were created (Mayers, 2013). To satisfy the assumption of linearity, the interaction outcome (when predictors are transformed) should not be significant. Table 8.7a shows that the linearity assumption is satisfied.

d. multi-collinearity: Table 8.7.b shows that this assumption is also satisfied.

Table 8.7a linearity assumption test

<table>
<thead>
<tr>
<th>Variables in the Equation</th>
<th>B</th>
<th>S.E.</th>
<th>Wald</th>
<th>df</th>
<th>Sig.</th>
<th>Exp(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse nor mal</td>
<td>-11868.110</td>
<td>219811.156</td>
<td>.003</td>
<td>1</td>
<td>.957</td>
<td>.000</td>
</tr>
<tr>
<td>Step 1a efficacyProg</td>
<td>-2406.614</td>
<td>119388.197</td>
<td>.000</td>
<td>1</td>
<td>.984</td>
<td>.000</td>
</tr>
<tr>
<td>taskValue</td>
<td>-11322.795</td>
<td>204201.983</td>
<td>.003</td>
<td>1</td>
<td>.956</td>
<td>.000</td>
</tr>
<tr>
<td>Variable</td>
<td>Unstandardized Coef</td>
<td>Standardized Coef</td>
<td>p-value</td>
<td>Sig</td>
<td></td>
<td>-----</td>
</tr>
<tr>
<td>reverse</td>
<td>Model</td>
<td>Coefficients</td>
<td>logidentity</td>
<td>motivation</td>
<td>senseBelonging</td>
<td>taskValue</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>--------------</td>
<td>-------------</td>
<td>------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
</tbody>
</table>
### 8.7.1.1 Part 1: predicting liminal or post-liminal space considering both students’ self-evaluations and the other affective variables

#### 8.7.1.1.1 Model for the non-reversed group

For the group that the questionnaire was given first, binary logistic regression (Tables 8.7.1.1-8.7.1.13) was used to predict an outcome of liminality among 40 participants. The final model was able to explain 78.6% of the variance. The model was found to fit the data adequately (Hosmer and Lemeshow's $x^2 = 3.498$, $p = .899$) and was able to predict liminality status (Omnibus $x^2(4) = 35.401$, $p < .000$). Overall, the model was able to correctly predict 90.0% of all cases. Four predictors were included in the model using the enter method: self-efficacy, motivation, identity and self-evaluation. Two of them significantly predicted liminality status: motivation and self-evaluation on the test (squared Wald statistics are displayed in the table). Assumptions for linearity and multicollinearity were satisfied.
Logistic Regression

Table 8.7.1.1 Case Processing Summary

<table>
<thead>
<tr>
<th>reverse</th>
<th>Unweighted Cases(a)</th>
<th>N</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>Selected Cases</td>
<td>40</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>Included in Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Missing Cases</td>
<td>0</td>
<td>.0</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>40</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>Unselected Cases</td>
<td>0</td>
<td>.0</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>40</td>
<td>100.0</td>
</tr>
</tbody>
</table>

\(a\). If weight is in effect, see classification table for the total number of cases.

Table 8.7.1.2 Dependent Variable Encoding

<table>
<thead>
<tr>
<th>Reverse</th>
<th>Original Value</th>
<th>Internal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>Yes</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Block 0: Beginning Block

Table 8.7.1.3 Iteration History

<table>
<thead>
<tr>
<th>reverse</th>
<th>Iteration</th>
<th>-2 Log likelihood</th>
<th>Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Constant</td>
</tr>
<tr>
<td>normal</td>
<td>Step 0</td>
<td>1</td>
<td>55.051(^b,c)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>55.051</td>
</tr>
</tbody>
</table>

\(a\). Constant is included in the model.
b. Initial \(-2\) Log Likelihood: 55.051 for split file reverse = normal

c. Estimation terminated at iteration number 2 because parameter estimates changed by less than .001 for split file reverse = normal.

### Table 8.7.1.4 Classification table

<table>
<thead>
<tr>
<th>Reverse</th>
<th>Observed</th>
<th>Predicted</th>
<th>Percentage Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Liminal</td>
<td>No</td>
</tr>
<tr>
<td>normal</td>
<td>Step 0</td>
<td>Liminal</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overall Percentage</td>
<td></td>
</tr>
</tbody>
</table>

a. Constant is included in the model.
b. The cut value is .500

### Table 8.7.1.5 Variables in the Equation

<table>
<thead>
<tr>
<th>reverse</th>
<th>B</th>
<th>S.E.</th>
<th>Wald</th>
<th>df</th>
<th>Sig.</th>
<th>Exp(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>Step 0 a</td>
<td>Constant</td>
<td>.201</td>
<td>.318</td>
<td>.399</td>
<td>1</td>
</tr>
</tbody>
</table>

a. Variable(s) entered on step 1: efficacyProg, motivation, identity, self-evaluation.

### Table 8.7.1.6 Variables not in the Equation

<table>
<thead>
<tr>
<th>reverse</th>
<th>Score</th>
<th>df</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>Step 0</td>
<td>Variables</td>
<td>efficacyProg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>motivation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>identity</td>
</tr>
<tr>
<td>reverse</td>
<td>Iteration</td>
<td>-2 Log likelihood</td>
<td>Coefficients</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>-------------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Constant</td>
</tr>
<tr>
<td>normal</td>
<td>Step 1</td>
<td>1</td>
<td>29.119c,d</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>22.251</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>20.093</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>19.679</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>19.651</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>19.650</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>19.650</td>
</tr>
</tbody>
</table>

a. Method: Enter

b. Constant is included in the model.

c. Initial -2 Log Likelihood: 55.051 for split file reverse = normal

d. Estimation terminated at iteration number 7 because parameter estimates changed by less than .001 for split file reverse = normal.

e. Initial -2 Log Likelihood: 101.572 for split file reverse = not

f. Estimation terminated at iteration number 5 because parameter estimates changed by less than .001 for split file reverse = not.
### Table 8.7.1.8 Omnibus Tests of Model Coefficients

<table>
<thead>
<tr>
<th>Reverse</th>
<th>Chi-square</th>
<th>df</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Step 1</td>
<td>35.401</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Block</td>
<td>35.401</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>35.401</td>
<td>4</td>
</tr>
</tbody>
</table>

### Table 8.7.1.9 Model Summary

<table>
<thead>
<tr>
<th>Reverse</th>
<th>Step</th>
<th>-2 Log likelihood</th>
<th>Cox &amp; Snell R Square</th>
<th>Nagelkerke R Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>1</td>
<td>19.650a</td>
<td>.587</td>
<td>.786</td>
</tr>
</tbody>
</table>

*a. Estimation terminated at iteration number 7 because parameter estimates changed by less than .001 for split file reverse = normal.*

### Table 8.7.1.10 Hosmer and Lemeshow Test

<table>
<thead>
<tr>
<th>Reverse</th>
<th>Step</th>
<th>Chi-square</th>
<th>df</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>1</td>
<td>3.498</td>
<td>8</td>
<td>.899</td>
</tr>
</tbody>
</table>

### Table 8.7.1.11 Contingency Table for Hosmer and Lemeshow Test

<table>
<thead>
<tr>
<th>Reverse</th>
<th>Liminal = No</th>
<th>Liminal = Yes</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Observed</td>
<td>Expected</td>
<td>Observed</td>
</tr>
<tr>
<td>normal</td>
<td>Step 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
<td>3.977</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td>3.908</td>
</tr>
</tbody>
</table>
Table 8.7.1.12 Classification Table

<table>
<thead>
<tr>
<th>reverse</th>
<th>Observed</th>
<th>Predicted</th>
<th>Liminal</th>
<th>Percentage Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>normal</td>
<td>Step 1</td>
<td>Liminal</td>
<td>No</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overall Percentage</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. The cut value is .500

Table 8.7.1.13 Variables in the Equation

<table>
<thead>
<tr>
<th>reverse</th>
<th>B</th>
<th>S.E.</th>
<th>Wald</th>
<th>df</th>
<th>Sig.</th>
<th>Exp(B)</th>
<th>95% C.I.for EXP(B)</th>
<th>95% C.I.for EXP(B)</th>
<th>Lower</th>
<th>Upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>norm al</td>
<td>Step 1</td>
<td>efficacyProg</td>
<td>-2.753</td>
<td>1.427</td>
<td>3.722</td>
<td>1</td>
<td>.054</td>
<td>.064</td>
<td>.004</td>
<td>1.045</td>
</tr>
<tr>
<td></td>
<td>motiva tion</td>
<td>3.162</td>
<td>1.494</td>
<td>4.478</td>
<td>1</td>
<td>.034</td>
<td>23.625</td>
<td>1.263</td>
<td>442.029</td>
<td></td>
</tr>
</tbody>
</table>
8.7.1.1.2 Model for the reversed group

For the group that the questionnaire was given after the test, binary logistic regression (Tables 8.7.1.14-8.7.1.26) was used to predict an outcome of liminality among 83 participants. The final model was able to explain 39.1% of the variance. The model was found to fit the data adequately (Hosmer and Lemeshow's $x^2 = 8.233$, $p = .411$) and was able to predict liminality status (Omnibus $x^2(6) = 26.830$, $p < .000$). Overall, the model was able to predict 79.5% of all cases correctly. Six predictors were included in the model using the enter method: self-evaluation, self-efficacy, task value, sense of belonging, motivation, and identity. One of them successfully predicted liminality status: self-evaluation on the test (squared Wald statistics are displayed in the table).

**Logistic Regression**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>Standard Error</th>
<th>t-Value</th>
<th>p-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>identity</td>
<td>-1.946</td>
<td>1.099</td>
<td>1.315</td>
<td>.077</td>
</tr>
<tr>
<td>self-evaluation</td>
<td>-1.075</td>
<td>.405</td>
<td>2.675</td>
<td>.008</td>
</tr>
<tr>
<td>Constant</td>
<td>16.329</td>
<td>8.050</td>
<td>2.027</td>
<td>.154</td>
</tr>
</tbody>
</table>

a. Variable(s) entered on step 1: efficacyProg, motivation, identity, self-evaluation.

**Table 8.7.1.14 Case Processing Summary**

<table>
<thead>
<tr>
<th>reverse</th>
<th>Unweighted Cases&lt;sup&gt;a&lt;/sup&gt;</th>
<th>N</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>Selected Cases</td>
<td>Included in Analysis</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Missing Cases</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>Total</td>
<td>83</td>
</tr>
<tr>
<td>Unselected Cases</td>
<td>0</td>
<td>.0</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-----</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>83</td>
<td>100.0</td>
<td></td>
</tr>
</tbody>
</table>

a. If weight is in effect, see classification table for the total number of cases.

### Table 8.7.1.15 Dependent Variable Encoding

<table>
<thead>
<tr>
<th>Dependent Variable Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse</td>
</tr>
<tr>
<td>not</td>
</tr>
<tr>
<td>Yes</td>
</tr>
</tbody>
</table>

### Block 0: Beginning Block

### Table 8.7.1.16 Iteration History

<table>
<thead>
<tr>
<th>Iteration History(^a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse</td>
</tr>
<tr>
<td>not</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

a. Constant is included in the model.

d. Initial -2 Log Likelihood: 101.572 for split file reverse = not

e. Estimation terminated at iteration number 3 because parameter estimates changed by less than .001 for split file reverse = not.
### Table 8.7.1.17 Classification Table

<table>
<thead>
<tr>
<th>reverse</th>
<th>Observed</th>
<th>Predicted</th>
<th>Liminal</th>
<th>Percentage Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>not</td>
<td>Step 0</td>
<td>Liminal</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>0</td>
</tr>
<tr>
<td>Overall Percentage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Constant is included in the model.

b. The cut value is .500

### Table 8.7.1.18 Variables in the Equation

<table>
<thead>
<tr>
<th>reverse</th>
<th>B</th>
<th>S.E.</th>
<th>Wald</th>
<th>df</th>
<th>Sig.</th>
<th>Exp(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>Step 0</td>
<td>Constant</td>
<td>.842</td>
<td>.239</td>
<td>12.373</td>
<td>.000</td>
</tr>
</tbody>
</table>

a. Variable(s) entered on step 1: efficacyProg, motivation, identity, self-evaluation, taskValue, senseBelonging.

### Table 8.7.1.19 Variables not in the Equation

<table>
<thead>
<tr>
<th>reverse</th>
<th>Score</th>
<th>df</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td>efficacyProg</td>
<td>11.524</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>motivation</td>
<td>.690</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>identity</td>
<td>4.123</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>self-evaluation</td>
<td>20.634</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>taskValue</td>
<td>1.484</td>
<td>1</td>
</tr>
</tbody>
</table>
### Table 8.7.1.20 Iteration History

<table>
<thead>
<tr>
<th>reverse</th>
<th>Iteration</th>
<th>-2 Log likelihood</th>
<th>Coefficients</th>
<th>Const</th>
<th>efficacyP</th>
<th>motivation</th>
<th>identity</th>
<th>self-evaluation</th>
<th>taskValue</th>
<th>senseBelonging</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>Step 1</td>
<td>77.905^c,f</td>
<td>3.234, -.344, -.274, -.068, -.381, .279, .313</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>74.895</td>
<td>4.447, -.495, -.405, -.070, -.538, .424, .431</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>74.743</td>
<td>4.775, -.537, -.440, -.067, -.584, .472, .456</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>74.742</td>
<td>4.797, -.540, -.442, -.067, -.587, .476, .457</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>74.742</td>
<td>4.797, -.540, -.442, -.067, -.587, .476, .457</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Method: Enter

b. Constant is included in the model.

c. Initial -2 Log Likelihood: 55.051 for split file reverse = normal

d. Estimation terminated at iteration number 7 because parameter estimates changed by less than .001 for split file reverse = normal.

e. Initial -2 Log Likelihood: 101.572 for split file reverse = not

f. Estimation terminated at iteration number 5 because parameter estimates changed by less than .001 for split file reverse = not.

### Table 8.7.1.21 Omnibus Tests of Model Coefficients

<table>
<thead>
<tr>
<th>Omnibus Tests of Model Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Not</td>
</tr>
<tr>
<td>Block</td>
</tr>
<tr>
<td>reverse</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>not</td>
</tr>
</tbody>
</table>

b. Estimation terminated at iteration number 5 because parameter estimates changed by less than .001 for split file reverse = not.

Table 8.7.1.23 Hosmer and Lemeshow Test

<table>
<thead>
<tr>
<th>reverse</th>
<th>Step</th>
<th>Chi-square</th>
<th>df</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>1</td>
<td>8.233</td>
<td>8</td>
<td>.411</td>
</tr>
</tbody>
</table>

Table 8.7.1.24 Contingency Table for Hosmer and Lemeshow Test

<table>
<thead>
<tr>
<th>reverse</th>
<th>Liminal = No</th>
<th>Liminal = Yes</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Observed</td>
<td>Expected</td>
<td>Observed</td>
</tr>
<tr>
<td></td>
<td>Observed</td>
<td>Expected</td>
<td>Observed</td>
</tr>
<tr>
<td>not</td>
<td>Step 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>6.553</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>5.032</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4.013</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3.022</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2.288</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>1.286</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1.036</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>.776</td>
<td>7</td>
</tr>
</tbody>
</table>
Table 8.7.1.25 Classification Table

<table>
<thead>
<tr>
<th>reverse</th>
<th>Observed</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Liminal</td>
<td>Percentage Correct</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>not</td>
<td>Step 1</td>
<td>Liminal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overall Percentage</td>
<td>79.5</td>
</tr>
</tbody>
</table>

a. The cut value is .500

Table 8.7.1.26 Variables in the Equation

<table>
<thead>
<tr>
<th>Variables in the Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>not</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
8.7.1.2 Part 2: predicting liminal or post-liminal space considering only the affective factors apart from the self-evaluation

8.7.1.2.1 Model for the non-reversed group

For the group that the questionnaire was given first, binary logistic regression (Tables 8.7.1.27-8.7.1.35) was used to predict an outcome of liminality among 40 participants. The final model was able to explain 59.2% of the variance. The model was found to fit the data adequately (Hosmer and Lemeshow's $x^2 = 4.533$, $p = .806$) and was able to predict liminality status (Omnibus $x^2(4) = 23.383$, $p < .000$). Overall, the model was able to correctly predict 82.5% of all cases. Four predictors were included in the model using the enter method: self-efficacy, sense of belonging, motivation and identity. Three of them successfully predicted liminality status: self-efficacy, identity and motivation (squared Wald statistics are displayed in the table). Assumptions for linearity and multicollinearity were satisfied.

<table>
<thead>
<tr>
<th>constant</th>
<th>4.79</th>
<th>7</th>
<th>1.77</th>
<th>2</th>
<th>7.33</th>
<th>1</th>
<th>1</th>
<th>.007</th>
<th>121.17</th>
<th>3</th>
<th>.791</th>
<th>3.557</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sense Belonging</td>
<td>.457</td>
<td>.414</td>
<td>1.217</td>
<td>1</td>
<td>.270</td>
<td>1.579</td>
<td>.701</td>
<td>.701</td>
<td>3.557</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Variable(s) entered on step 1: efficacyProg, motivation, identity, self-evaluation, taskValue, senseBelonging.

Table 8.7.1.27 Case Processing Summary

<p>| Case Processing Summary |  |
|---|---|---|---|---|---|---|---|---|---|
| reverse | Unweighted Cases | N | Percent |  |
| normal | Selected Cases | Included in Analysis | 40 | 100.0 |  |
| | Missing Cases | 0 | .0 |  |
| | Total | 40 | 100.0 |  |
| Unselected Cases | 0 | .0 |  |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>40</td>
<td>100.0</td>
</tr>
<tr>
<td>not Selected Cases</td>
<td>Included in Analysis</td>
<td>83</td>
</tr>
<tr>
<td>Missing Cases</td>
<td>0</td>
<td>.0</td>
</tr>
<tr>
<td>Total</td>
<td>83</td>
<td>100.0</td>
</tr>
<tr>
<td>Unselected Cases</td>
<td>0</td>
<td>.0</td>
</tr>
<tr>
<td>Total</td>
<td>83</td>
<td>100.0</td>
</tr>
</tbody>
</table>

a. If weight is in effect, see classification table for the total number of cases.

### Table 8.7.1.28 Dependent Variable Encoding

<table>
<thead>
<tr>
<th>Dependent Variable Encoding</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse</td>
<td>Original Value</td>
<td>Internal Value</td>
</tr>
<tr>
<td>normal</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
</tr>
<tr>
<td>not</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>1</td>
</tr>
</tbody>
</table>

### Block 0: Beginning Block

### Table 8.7.1.28 Classification Table

<table>
<thead>
<tr>
<th>Classification Table&lt;sup&gt;a,b&lt;/sup&gt;</th>
<th>Predicted</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse</td>
<td>Observed</td>
<td>Percentage Correct</td>
</tr>
<tr>
<td>normal</td>
<td>Liminal</td>
<td>No</td>
</tr>
<tr>
<td>Step 0</td>
<td>Liminal</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Overall Percentage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td>Liminal</td>
<td>No</td>
</tr>
<tr>
<td>Step 0</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Overall Percentage</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Constant is included in the model.
b. The cut value is .500

### Table 8.7.1.29 Variables in the Equation

<table>
<thead>
<tr>
<th>Variables in the Equation</th>
<th></th>
<th></th>
</tr>
</thead>
</table>

287
### Table 8.7.1.30 Variables not in the Equation

<table>
<thead>
<tr>
<th>Reverse</th>
<th>Step 0</th>
<th>Variables</th>
<th>Score</th>
<th>df</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td></td>
<td>efficacyProg</td>
<td>7.867</td>
<td>1</td>
<td>.005</td>
</tr>
<tr>
<td></td>
<td></td>
<td>identity</td>
<td>4.311</td>
<td>1</td>
<td>.038</td>
</tr>
<tr>
<td></td>
<td></td>
<td>motivation</td>
<td>.000</td>
<td>1</td>
<td>.991</td>
</tr>
<tr>
<td></td>
<td></td>
<td>senseBelonging</td>
<td>1.065</td>
<td>1</td>
<td>.302</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overall Statistics</td>
<td>15.464</td>
<td>4</td>
<td>.004</td>
</tr>
<tr>
<td>not</td>
<td></td>
<td>efficacyProg</td>
<td>11.524</td>
<td>1</td>
<td>.001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>identity</td>
<td>4.123</td>
<td>1</td>
<td>.042</td>
</tr>
<tr>
<td></td>
<td></td>
<td>motivation</td>
<td>.690</td>
<td>1</td>
<td>.406</td>
</tr>
<tr>
<td></td>
<td></td>
<td>senseBelonging</td>
<td>1.601</td>
<td>1</td>
<td>.206</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overall Statistics</td>
<td>12.367</td>
<td>4</td>
<td>.015</td>
</tr>
</tbody>
</table>

### Block 1: Method = Enter

### Table 8.7.1.31 Omnibus Tests of Model Coefficients

<table>
<thead>
<tr>
<th>Omnibus Tests of Model Coefficients</th>
<th>Reverse</th>
<th>Step 1</th>
<th>Chi-square</th>
<th>df</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Step</td>
<td>23.383</td>
<td>4</td>
<td>.000</td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>23.383</td>
<td>4</td>
<td>.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not</td>
<td>Step</td>
<td>13.713</td>
<td>4</td>
<td>.008</td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>13.713</td>
<td>4</td>
<td>.008</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 8.7.1.32 Model Summary

<table>
<thead>
<tr>
<th>reverse</th>
<th>Step</th>
<th>-2 Log likelihood</th>
<th>Cox &amp; Snell R Square</th>
<th>Nagelkerke R Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>1</td>
<td>31.668&lt;sup&gt;a&lt;/sup&gt;</td>
<td>.443</td>
<td>.592</td>
</tr>
<tr>
<td>not</td>
<td>1</td>
<td>87.859&lt;sup&gt;b&lt;/sup&gt;</td>
<td>.152</td>
<td>.216</td>
</tr>
</tbody>
</table>

a. Estimation terminated at iteration number 6 because parameter estimates changed by less than .001 for split file reverse = normal.

b. Estimation terminated at iteration number 5 because parameter estimates changed by less than .001 for split file reverse = not.

### Table 8.7.1.33 Hosmer and Lemeshow Test

<table>
<thead>
<tr>
<th>reverse</th>
<th>Step</th>
<th>Chi-square</th>
<th>df</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>1</td>
<td>4.533</td>
<td>8</td>
<td>.806</td>
</tr>
<tr>
<td>not</td>
<td>1</td>
<td>10.340</td>
<td>8</td>
<td>.242</td>
</tr>
</tbody>
</table>

### Table 8.7.1.34 Classification Table

<table>
<thead>
<tr>
<th>reverse</th>
<th>Observed</th>
<th>Predicted</th>
<th>Percentage Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Liminal</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>normal</td>
<td>Step 1</td>
<td>Liminal</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Liminal</td>
<td>Overall Percentage</td>
</tr>
<tr>
<td>not</td>
<td>Step 1</td>
<td>Liminal</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Liminal</td>
<td>Overall Percentage</td>
</tr>
</tbody>
</table>

a. The cut value is .500

### Table 8.7.1.35 Variables in the Equation

<table>
<thead>
<tr>
<th>reverse</th>
<th>B</th>
<th>S.E.</th>
<th>Wald</th>
<th>df</th>
<th>Sig.</th>
<th>Exp(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>Step 1&lt;sup&gt;a&lt;/sup&gt;</td>
<td>efficacyProg</td>
<td>-2.458</td>
<td>.988</td>
<td>6.183</td>
<td>1.013</td>
</tr>
</tbody>
</table>

<sup>a</sup> Estimation terminated at iteration number 6 because parameter estimates changed by less than .001 for split file reverse = normal.
<table>
<thead>
<tr>
<th></th>
<th>Motivation</th>
<th>.701</th>
<th>1.505</th>
<th>6.050</th>
<th>1</th>
<th>.014</th>
<th>40.491</th>
</tr>
</thead>
<tbody>
<tr>
<td>senseBelonging</td>
<td>-1.176</td>
<td>1.146</td>
<td>1.053</td>
<td>1</td>
<td>.305</td>
<td>.308</td>
<td></td>
</tr>
<tr>
<td>Constant</td>
<td>11.610</td>
<td>5.713</td>
<td>4.129</td>
<td>1</td>
<td>.042</td>
<td>110203.14</td>
<td></td>
</tr>
<tr>
<td>Step 1</td>
<td>efficacyProg</td>
<td>-.900</td>
<td>.324</td>
<td>7.703</td>
<td>1</td>
<td>.006</td>
<td>.407</td>
</tr>
<tr>
<td>Identity</td>
<td>-.155</td>
<td>.309</td>
<td>.252</td>
<td>1</td>
<td>.616</td>
<td>.856</td>
<td></td>
</tr>
<tr>
<td>Motivation</td>
<td>.017</td>
<td>.365</td>
<td>.002</td>
<td>1</td>
<td>.964</td>
<td>1.017</td>
<td></td>
</tr>
<tr>
<td>senseBelonging</td>
<td>.258</td>
<td>.337</td>
<td>.587</td>
<td>1</td>
<td>.444</td>
<td>1.294</td>
<td></td>
</tr>
<tr>
<td>Constant</td>
<td>4.664</td>
<td>1.562</td>
<td>8.918</td>
<td>1</td>
<td>.003</td>
<td>106.035</td>
<td></td>
</tr>
</tbody>
</table>

a. Variable(s) entered on step 1: efficacyProg, identity, motivation, senseBelonging.

### 8.8 Conclusion

The chapter presented here illustrated the results of the third phase of the research and specifically, the first part of this phase. All the results of the statistical tests were presented in order to answer the 5th and 6th and the 7th research questions of the study referring to students’ affective dimensions of learning and if and how these are impacted in liminality.

The results highlight that the students in liminal space experience statistically significant lower levels of self-efficacy and computer science identity than the students in the post-liminal space. For the other variables, although the levels of sense of belonging and motivation were lower for the students in liminal space than the students in post-liminal space, these differences were not statistically significant.

When the impact of troublesome knowledge was examined, it was evident that within the groups, students who completed the affective questionnaire after the programming tasks
experienced statistically significant lower levels of self-efficacy, motivation and sense of belonging than the students who completed the questionnaire before the test.

In the final section of this chapter, different affective models were examined for identifying students who may be in liminal space. The most successful model takes into consideration the following variables: self-efficacy, motivation, identity and self-evaluation and correctly predicted the liminal state of 90% of all cases (non-reverse group). The second model (reverse group) uses six predictors, self-evaluation, self-efficacy, task value, sense of belonging, motivation, and identity and correctly predicted 79.5% of all cases. The final model (non-reverse group) uses four predictors, self-efficacy, motivation, identity and sense of belonging and correctly predicted 82.5% of all cases.

A detailed discussion of the aforementioned results is presented in chapter 10 while the next chapter presents the results concerning the second part of the third phase, which focuses on uncovering students’ misconceptions in functions.
Chapter 9 - Results: Phase 3b – Students’ misconceptions in functions

The current chapter presents the data analysis and the results of the second part of the final phase of the research and specifically the misconceptions held by students in functions. To identify students’ misconceptions in functions, the students’ responses in the programming tasks were qualitatively analysed. The chapter begins by presenting the interrater reliability and then proceeds with presenting the analysis of each task, the errors and the misconceptions discovered.

9.1 Common misconceptions when students are in liminal spaces

The aim of the programming tasks analysis conducted in this phase of the research was to discover common errors that students make in programming and to uncover misconceptions that may be held by students in functions. To this end, each of the programming tasks was analysed independently for each student following the guidelines of Shah et al. (2017), and the errors that were found were written down on a list, generalised, and then categorised into groups. It is important to mention here that no misconceptions were found for students that belonged to the post-liminal group.

9.1.1 Interrater reliability

To ensure the validity and reliability of the interpretation of the misconceptions, two researchers were involved in the process: as the first researcher, I produced the coder manual with the categories (misconceptions) which was used by the second researcher to categorise the students’ errors again. The interrater reliability of Cohen’s kappa was calculated (Table...
and a substantial agreement was found: $k=0.675$, $p < .001$ (approximate 95% confidence interval on Kappa: 0.612 - 0.737). Only categories that both researchers agreed upon are presented in this chapter.

<table>
<thead>
<tr>
<th>Symmetric Measures</th>
<th>Value</th>
<th>Asymptotic Standard Error$^a$</th>
<th>Approximate $T^b$</th>
<th>Approximate Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure of Agreement</td>
<td>Kappa</td>
<td>.675</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N of Valid Cases</td>
<td>306</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Not assuming the null hypothesis.
b. Using the asymptotic standard error assuming the null hypothesis.

9.1.2 Investigating students’ errors per task

9.1.2.1 Task 1 - Basic Understanding of function concepts: concept matching exercise

The first exercise was a multiple-choice exercise in which the students were asked to match a concept, e.g. parameters with the appropriate annotated fragment in the code. The following figure represents the task given to students who were called to answer the following question: “The following program has 7 lines of code. Some parts are underlined and annotated with a letter. Match each of the following concepts with the appropriate letter in the code. An example is demonstrated below”.
The table presented below demonstrates students’ mistakes and the percentage of students that demonstrate these mistakes.

Table 9.2.1 Percentages of students per error

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Correct Answers: 48.7%</th>
<th>Incorrect Answers: 58.1%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Percentage of total students demonstrated the error</td>
<td>Percentage of students that did this error among those who did not answer the task correctly</td>
</tr>
</tbody>
</table>

Question 1 (parameters)

<table>
<thead>
<tr>
<th>Error 1:</th>
<th>13.11%</th>
<th>51.6%</th>
</tr>
</thead>
<tbody>
<tr>
<td>the student confuses parameters with arguments</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error 2:</th>
<th>7.37%</th>
<th>29%</th>
</tr>
</thead>
<tbody>
<tr>
<td>the student confuses parameters with calling a function</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error 3:</th>
<th>3.27%</th>
<th>12.9%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
the student confuses parameters with the function definition

<table>
<thead>
<tr>
<th>Question 2 (arguments)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Error 4:</td>
<td></td>
</tr>
<tr>
<td>the student confuses arguments with parameters</td>
<td>16.39%</td>
</tr>
<tr>
<td>Error 5:</td>
<td></td>
</tr>
<tr>
<td>the student confuses arguments with calling a function</td>
<td>13.93%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question 3 (calling a function)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Error 6:</td>
<td></td>
</tr>
<tr>
<td>the student confuses calling a function with function definition</td>
<td>23.7%</td>
</tr>
<tr>
<td>Error 7:</td>
<td></td>
</tr>
<tr>
<td>the student confuses calling a function with arguments</td>
<td>8.19%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question 5 (function definition)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Error 8:</td>
<td></td>
</tr>
<tr>
<td>the student confuses function definition with calling</td>
<td>11.47%</td>
</tr>
<tr>
<td>Error 9:</td>
<td></td>
</tr>
<tr>
<td>the student confuses function definition with parameters</td>
<td>5.73%</td>
</tr>
<tr>
<td>Error 10:</td>
<td></td>
</tr>
<tr>
<td>the student confuses function definition with arguments</td>
<td>4.91%</td>
</tr>
</tbody>
</table>

As table 9.2.1 indicates, less than half of the students answered the whole task correctly (48.7%). The most common errors that students made concern with mismatching “calling a function” with “function definition” and “parameters” with “arguments”. Specifically, 51.6% of the students made the latter error. Additionally, students confused calling a function with arguments with 42.5% of the students making this error. Finally, 64.4% of the students mismatched “calling a function” with “function definition”.
9.1.2.2 Task 2 - Knowledge of syntax and semantics in functions: find the errors exercise

The second exercise tested students’ knowledge in functions and particularly with handling parameters and arguments. Students were given the following exercise, and they were asked to identify the errors and to correct them. As it can be seen in Table 9.2.2, only 5.69% of the students identified all the errors in the code. The rest of the students identified most of the syntax errors, but only a few of them corrected errors focused on the arguments and parameters.

```python
1. def calculate age (year):
2.     students_age = 2018 - year
3.
4. def main()
5.     yearofbirth=int(input("type the year of birth:"))
6.     age = calculate_age ()
7.     print ("the student's age is " age)
```

**Figure 6 Task 2**

<table>
<thead>
<tr>
<th>Task 2</th>
<th>Correct Answers: 5.69%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Incorrect Answers: 94.31%</td>
</tr>
<tr>
<td></td>
<td>Percentage of total students</td>
</tr>
</tbody>
</table>

| Line 1: | Correct: 78.8% |
|---------| Incorrect: 21.2% |

| Line 2: Misconception1 | Correct: 80.4% |
|------------------------| Incorrect: 19.6% |

| Line 3: | Correct: 16.2% |
|---------| Incorrect: 73.9% |

| Line 4: | Correct: 57.7% |
From the students’ responses, two misconceptions were uncovered: the first is that parameters must be explicitly defined to be used inside the function which is somehow linked with the second misconception which is that parameters and arguments must have the same name. Examples of students’ answers demonstrating these misconceptions are depicted in the following table:

<table>
<thead>
<tr>
<th>Student 1 – Misconception 1</th>
<th>“Second line: year can’t be used as it has no value”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 2 – Misconception 2</td>
<td>“Fifth line: yearofbirth should be named year and then use it in the next statement”</td>
</tr>
<tr>
<td>Student 3 – Misconception 2</td>
<td>“age=calculate_age(year)”</td>
</tr>
</tbody>
</table>
9.1.2.3 Task 3 - Code comprehension and tracing with functions

The third task was about code comprehension and tracing. The students were given a program with two functions and the main function, and they were asked to indicate the program’s purpose and output. In comparison with the other exercises, the students did quite well in this exercise. In total, 78.8% of the students answered this task correctly.

```python
1. def decision (type):
2.     if type = "expensive":
3.         print "unable to buy!"
4.     else:
5.         print "you can buy the product!"
6.
7. def price_evaluation (price):
8.     if price < 10:
9.         eval = "cheap"
10.    elif price < 20:
11.        eval = "normal"
12.    else:
13.        eval = "expensive"
14.    return eval
15.
16. def main()
17.    cost=int(input("type the product’s price:"))
18.    evaluation = price_evaluation (cost)
19.    decision (evaluation)
```

*Figure 7 Task 3*

9.1.2.4 Task 4 - Parsons puzzle with functions

The fourth task was a Parson’s puzzle task where the students were given a description about a program’s purpose and a set of unordered code statements, and they were asked to put the statements in the right order for the program to achieve its goal. In total, 52% of students answered the task correctly and put all the statements in the right order, while 48% of the students did not. From the number of students that did not answer the task correctly, a 72.8% misplaced the return statement under the call of the function in the main function or under
the print statement in the main function. This led me to the conclusion that some students may think that return statements can be placed under the call of the function to return and print the function’s output which was also evident in the last exercise.

```python
1. def output (grade):
2.     result = "pass"
3.     result = "fail"
4.     print output(mark)
5.     return result
6. def main ():
7.     if grade > 12:
8.         mark = int(input("your mark:"))
9.     else:
```

*Figure 8 Task 4*

Table 9.2.4 Task 4 Misconceptions

<table>
<thead>
<tr>
<th>Task 4</th>
<th>Correct Answers: 52.0%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misconception3</td>
<td>Incorrect Answers: 48.0%</td>
</tr>
<tr>
<td>Return statement can be used after the call to the function to return and print the function’s output</td>
<td>72.8% of those that did not answer the task correctly</td>
</tr>
</tbody>
</table>

An example that demonstrates this misconception is provided in the following figure:
9.1.2.5 Task 5 - Advance understanding - Fill in the blanks

The fifth task required the students to understand how a program works in order to fill in the blanks correctly. There were three blanks that needed to be filled: Blank A was a parameter missing from the function definition, Blank B was a variable that holds the result that the function returns, and the Blank C was the argument in the function call in the main function. In total, 39.8% of students responded to this question correctly. The analysis of the students’ responses uncovered two new misconceptions and one that I have already uncovered in a previous task.

```python
def output(grade):
    if grade > 12:
        result = "pass"
    else:
        result = "fail"

def main():
    mark = int(input("insert grade:"))
    print(output(mark))
    return result
```

Figure 9 Example of students' errors

```python
def sum (_A__):
   _B__ = n1 + n2
   return result

def main ():
    number1 = int(input("type the first number"))
    number2 = int(input("type the second number"))
    print sum (_C__)
```

Figure 10 Task 5
Table 9.2.5 Task 5 Misconceptions

<table>
<thead>
<tr>
<th>Task 5</th>
<th>Correct Answers: 39.8%</th>
<th>Incorrect Answers: 60.2%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 5</td>
<td>Percentage of total students demonstrating the misconception</td>
<td>Percentage of students demonstrated this misconception among those who did not answer the task correctly</td>
</tr>
<tr>
<td>Misconception 2 (also evident on task 1)</td>
<td>15.4%</td>
<td>25.6%</td>
</tr>
<tr>
<td>Parameters and arguments should have the same name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misconception 4</td>
<td>4%</td>
<td>6.75%</td>
</tr>
<tr>
<td>Parameters should include the return variable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misconception 1 (also evident on task 1)</td>
<td>1.6%</td>
<td>4.7%</td>
</tr>
<tr>
<td>Parameters must be explicitly defined to be used inside the function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misconception 6</td>
<td>17%</td>
<td>28.3%</td>
</tr>
<tr>
<td>Return value must have the same name as the function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Misconception 5</td>
<td>12.1%</td>
<td>20.27%</td>
</tr>
<tr>
<td>Arguments should include the return variable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An example demonstrating this misconception 4 and 5 is depicted in the following figure:
In the final task, the students were asked to write their own programs. Specifically, I asked the students to write a program that calculates the area of a triangle, and I specifically instructed them to create a main function that reads the width and height of the triangle and then calls another function that calculates and returns the area back to the main function where the result is also printed. In total, 39.8% of students responded to this question correctly. The analysis of the students’ responses uncovered two new misconceptions and one that I have already identified in a previous task.

<table>
<thead>
<tr>
<th>Task 6</th>
<th>Correct: 39.8%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Incorrect: 60.2%</td>
</tr>
<tr>
<td>Misconception 6</td>
<td>Percentage of total students demonstrating the misconception</td>
</tr>
<tr>
<td>Return value must have the same name as the function</td>
<td>24.3%</td>
</tr>
<tr>
<td>Misconception 7</td>
<td>Local variables can be accessed from a function to another</td>
</tr>
</tbody>
</table>
Table 9.2.6 summarises the misconceptions that the study uncovered, the number of students that demonstrate the misconception (from those that did not answer the task correctly) and the task that illustrated the misconception.
## Table 9.2.6 Summary of Misconceptions found

<table>
<thead>
<tr>
<th>Misconceptions</th>
<th>Percentage of total students per task</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>The students think that return statement should be used after the call to the function to return and print the function’s output (M3)</td>
<td>47.96% - 6.5%</td>
<td>T4–T6</td>
</tr>
<tr>
<td>The students think that return value must have the same name with the function (M6)</td>
<td>24.3%</td>
<td>T6, T5</td>
</tr>
<tr>
<td>The students think that parameters must be explicitly defined to be used inside the function (M1)</td>
<td>19.6%</td>
<td>T2</td>
</tr>
<tr>
<td>The students think that parameters and arguments should have the same name (M2)</td>
<td>15.4% – 8.1%</td>
<td>T5–T2</td>
</tr>
<tr>
<td>The students think that arguments should have the return variable (M5)</td>
<td>12.1%</td>
<td>T5</td>
</tr>
<tr>
<td>The students think that local variables can be accessed from a function to another (M7)</td>
<td>10.6%</td>
<td>T6</td>
</tr>
<tr>
<td>The students think that calling a function prints the result (M8)</td>
<td>8.1%</td>
<td>T6</td>
</tr>
<tr>
<td>The students think that parameters should have the return variable (M4)</td>
<td>4% - 4%</td>
<td>T4–T6</td>
</tr>
</tbody>
</table>

### 9.2 Summary

The chapter presented the results of the second part of the final phase of the research. The chapter specifically attempted to provide an answer to the final research question of the study: What are the misconceptions of 16-year-old students in the area of functions? For
each task included in the programming test, the chapter presented the misconceptions uncovered as well as the percentage of students held this misconception.

In total, the study uncovered 8 misconceptions that students experience in functions. These misconceptions refer to the concepts of return values, and parameters. As the following chapter will discuss, some of these misconceptions have already been identified by other researchers while others have not yet been explored to my knowledge. It should also be noted that only students belonging in the liminal group demonstrated these misconceptions, whereas, for the post-liminal group, no misconceptions were evident. Following this chapter is the discussion of the third phase of the research, the results of which presented in this chapter and the previous one.
10 Chapter 10 - Discussion: Phase 3 – Affective dimensions of learning and misconceptions in threshold concepts

This section discusses the results presented in Chapter 8 and 9 regarding the third phase of the research. The research questions that this chapter endeavours to answer and discuss are the following:

a) Does being in liminal or post-liminal space affect students’ calibration, self-efficacy, task value, sense of belonging, motivation and computer science identity?

b) Does students’ encounter with troublesome knowledge impact their calibration, self-efficacy, task value, sense of belonging, motivation and computer science identity?

c) Can a model be used to predict students that are in liminal or post-liminal space?

d) What are the misconceptions of 16-year-old students in the area of functions?

The chapter starts by discussing students’ calibration ability in liminal and post-liminal space and then continues with the other affective dimensions of learning and the findings presented in Sections 8.3 through 8.5. The section continues with the regression models presented in Section 8.6, and then it finishes with the students’ misconceptions in functions that were discovered in this study.

10.1 Calibration and liminality

The first point of interest of the third phase was students’ calibration in the liminal and post-liminal space. As was discussed in the literature chapter, calibration refers to a meta-cognitive judgement of both students’ perceived performance and their actual performance on a task and is usually measured by the calibration accuracy and calibration bias. The role
of calibration in learning is critical as it is linked with self-regulation (Zimmerman, 1998), and thus, investigating students’ calibration when they are in liminal and post-liminal spaces is vital.

The results presented in Section 8.2 about students’ calibration bias and accuracy were interesting. The findings suggest that students in post-liminal space tend to underestimate their performance while students in liminal space slightly overestimate their performance. This is something that was expected, as research suggests that low-achieving students overrate their capabilities while high-achieving students underrate their own (Hacker et al., 2008). It is interesting to compare these findings with the results from Murphy and Tenenberg's (2005) study. Murphy and Tenenberg investigated computer science students’ calibration on a data structure course, and their results indicate a moderate negative correlation between performance and calibrations bias. Their findings align with this study, as students’ performance was also negatively correlated with calibration bias, corroborating the findings of Murphy and Tenenberg.

Considering the first research question of this phase regarding calibration ability, it can be argued that liminality does not seem to affect students’ calibration ability. Although students in liminal space tended to overestimate their performance, they were slightly better calibrated (though not significantly) than students in the post-liminal group. This result may be attributed to that students in the liminal space where students of different performance capabilities (range of performance: 13–79/100) who had not completely captured the relational level of the SOLO taxonomy. It is possible, therefore, that students with merit scores influenced the calibration ability of this group. Unfortunately, students with merit scores and students with low scores were assigned in the same group, that of liminal space,
as their performance was not a criterion for whether they are in liminal or post-liminal space (the criteria used for this distinction were listed in the methodology chapter and in section 3.2.3).

Nevertheless, the findings point out that even when students cross a threshold, their self-assessment judgements are not in line with their actual performance, which may have implications for their future career or subject choices. Hadwin and Webster (2012) posit that overconfidence may obstruct students from engaging with strategies that will help them achieve their learning goals, while under-confidence results in the over-employment of resources, both cognitively and affectively. From a pedagogical point of view, the findings suggest that teachers should not only focus on providing self-regulation opportunities for students who are in liminal space but also for students who seem to have passed the thresholds. In fact, self-regulation should be deeply incorporated into the programming class, and teachers should include self-evaluation in lessons while monitoring them for any inconsistencies. From students’ perspectives, this self-evaluation training will engage them with self-assessment opportunities that will help them evaluate their performance and capabilities more objectively, take control of their learning, and become self-regulated learners.

10.2 Self-efficacy in liminal and post-liminal space

The second point of interest in the third phase was students’ self-efficacy in programming. Specifically, this part of the research sought to investigate the students’ levels of self-efficacy when they are in the liminal and post-liminal space, as well as to explore the correlations that this motivational construct forms with the other affective dimensions of learning. To
this end, of particular interest was the examination of the role of troublesome knowledge and how it affects students’ perceptions of efficacy in programming when they are in the liminal and post-liminal space. The self-efficacy construct belongs to the affective dimensions of learning with a more cognitive structure, and as section 2.4.3 highlighted, it is well established in research that self-efficacy is important to students’ learning and performance.

The results presented in Chapter 8 show that students in the liminal space experienced statistically and significantly lower levels of self-efficacy in programming than their peers in the post-liminal space. This finding was expected, as generally, research about self-efficacy and performance has signified that students’ performance and self-efficacy are related and that students who demonstrate a higher level of self-efficacy are higher achievers. Self-efficacious students engage more with challenging tasks than other students, and they demonstrate better performance (Bandura & Schunk, 1981; Zimmerman & Kitsantas, 1997, 1999; Schunk et al., 1987; Salomon, 1984; Schunk, 1981; Multon et al., 1991). Similar studies that investigate factors that may influence students’ achievement in programming have highlighted the relationship-correlation between self-efficacy in programming and performance. For example, Wiedenbeck et al. (2004) advocate that previous experience in programming affects self-efficacy, which in turn impacts students’ success in this course.

The role of previous performance in programming and how it contributes to students’ self-efficacy in programming was also highlighted in Jegede's (2009) study, which found that students’ performance in programming courses significantly predicts their self-efficacy in programming.

Another important finding of this part of the research regards troublesome knowledge and whether students’ self-efficacy is influenced or not by it. The results suggest that when
students are confronted with troublesome knowledge, they experience an impact on their self-efficacy levels that is evident both in the liminal group and the post-liminal group. Considering students in liminal space, the self-efficacy level of students who did the programming test after the affective questionnaire was statistically and significantly higher than that of students who did the test before the questionnaire. Specifically, the difference in their self-efficacy mean score was 10.2. The same drop is evident for the students in the post-liminal group where the difference in their self-efficacy mean score was 9.34. It is therefore likely for students who are confronted with troublesome knowledge to experience a drop in their self-efficacy levels in programming, regardless of whether they are in liminal or in post-liminal space. It would be plausible to hypothesise that since students have crossed the thresholds, their self-efficacy levels would be stable and therefore uninfluenced by whether they confront troublesome knowledge or not. Contrary to expectation, this was not evident in this study. A possible reason for the change in the students’ self-efficacy levels in the post-liminal group may be attributed to their previous experience in programming. As Ramalingam’s and Wiedenbeck’s (1998) study suggests, students’ previous experience in programming impacts their self-efficacy, and thus, when given the programming test, students may recall these previous experiences, and this may subsequently impact their self-efficacy. To further support this statement, research suggests that self-efficacy is a dynamic construct, task specific and, thus, it changes through time as new information is assimilated (Gist and Mitchell, 1992).

This specific outcome presents an important finding for methodological and pedagogical approaches. From a methodological point of view, it seems that studies that are measuring students’ self-efficacy may produce different results depending on the time that the affective
questionnaire was distributed. From a pedagogical point of view, this finding suggests that teachers should design their assessment tests very carefully, as difficult questions may have an impact on students’ self-efficacy on the subject. It would be good practice for teachers to employ more strategies for enhancing students’ self-efficacy in programming after the administration of a test.

Considering the first research question of this phase referring to students’ self-efficacy, the results suggest that indeed there is a significant difference in the self-efficacy levels of students in the two groups, with students in the post-liminal group demonstrating higher levels of self-efficacy than students in the liminal group. This indicates that passing a threshold concept signifies a significant change in students’ self-efficacy in programming. In fact, this finding accords with this study’s earlier observations regarding the teachers’ experiences with threshold concepts. Specifically, most of the teachers in the second phase of the research indicated that once students understood these concepts, they started feeling more confident in programming and demonstrating better performance.

Considering the second research question of this phase referring to troublesome knowledge and self-efficacy, the results suggest that troublesome knowledge impact students’ self-efficacy. Although this finding was expected for students in the liminal group who experienced an uncertain situation in their learning, it is surprising that the same finding was evident for students in the post-liminal group. This result may suggest that previous experience with troublesome knowledge still affects students’ self-efficacy in programming. Another point may be that the post-liminal state is not a stable stage, an argument that it is discussed more in chapter 11.
10.3 Task value and liminal space

The third point of interest in this phase was about students’ task value when they are in liminal and post-liminal space. Task value, as was highlighted in the literature chapter, refers to the value students assign to computer programming. Specifically, in this part of the research, I was interested in investigating if the students’ levels of task value in computer programming was different between the two groups. In this investigation, I also explored if and how the task value levels varied when students experienced troublesome knowledge.

Examining the results presented in the 8th chapter, it is evident that the levels of task value for the two groups were not statistically different. In fact, the mean value for task value was quite high for both groups, which suggests that both the liminal and post-liminal group assigned a high value to their programming course and perceived it as useful for their future goals. Therefore, considering the threshold concept framework, it seems that task value is a construct that is not impacted by whether students are in liminal or post-liminal space. This may indicate that students who gave a high value to the course from the beginning maintained this level regardless of the difficulties they encountered. It is worth reiterating at this point that the students of this study had voluntarily selected computer science as an A-level course, which suggests that they selected the subject either to fulfil future career plans or for personal gains. Therefore, it is not surprising that students in both groups attributed high value to this course.

Furthermore, regarding the role of troublesome knowledge and its impact on students’ task value, the results showed that there was not a statistically significant difference in the task value score between the groups that did the tests before the questionnaire and after the
questionnaire, regardless of inclusion in either the liminal or post-liminal group. This indicates that troublesome knowledge does not have an impact on students’ task value in programming in comparison with what is evident in the self-efficacy construct. This finding was not anticipated since the literature suggests that students assign more value to tasks in which they perform well. This is because performing well in a task results in a positive effect that is then attached to the activity performed or, conversely, because assigning a lower value to challenging tasks is a possible way to maintain a high self-efficacy (Eccles et al., 1983). However, this study has been unable to demonstrate this point as no significant differences were depicted between students that were given the test first and students that were given the questionnaire first, which may indicate that when students have selected a course to fulfil personal goals, the value they assign to it is not affected by the difficulties they encounter.

Greene et al. (2004) suggested that when a subject is highly valued by the students in respect with their future goals, then the students are ready to work and study harder in order to master the course and to surpass their peers’ competence. Specifically, Greene et al. (2004) found a positive but indirect effect of perceived instrumentality (task value) on achievement with engagement as a mediator variable. This led the authors to support that instrumentality needs to be cultivated in schools and general education structures. On this basis, the fact that students’ task value in liminal space is high (mean = 5.50) is a positive indicator for mastering programming adeptly by adopting a mastery goal or by adopting a performance-approach goal.

Looking at the correlation tables, it is evident that the task value is correlated with all the other constructs under investigation. However, for the post-liminal group, the strongest correlation is formed between task value and sense of belonging ($r = .516$) and task value
and motivation \((r=.480)\). For the liminal group, the strongest correlation is formed between task value and self-efficacy \((r = .625)\). This suggests that for the post-liminal group, the value they assigned to the programming course was more closely related to their feelings of belonging in the class and the other way around, whereas for the liminal group, which still struggled with competence, the course value was more closely related to their efficacy beliefs. Green et al. (2017) suggest that task value is central to the willingness to be engaged with knowledge, while self-efficacy is central for progressing towards developing competence (Green et al., 2017). Therefore, it is not surprising that the task value is closely related to self-efficacy in the liminal group, which struggled to develop competence in the liminal space. The following section, which is about sense of belonging, motivation and identity, will provide more details about the relationship between task value and sense of belonging.

There is not much research about the effects of task value on conceptual changes, although indirect connections can be made. For instance, Johnson and Sinatra (2013) discuss the role of task value on conceptual change. They argue that different task value (utility, attainment) forces students to focus their attention on different facets of a task, thereby providing diverse ways by which new conceptions are linked to students’ task values. Specifically, their study found that students with utility task value experience the greatest amount of conceptual change compared to students with attainment values. The researchers interpret their findings by highlighting the importance of inducing students with task value for engagement and conceptual change. In the same line, Jones et al. (2015) postulate that the more value students place on a task, the more likely students are to engage with conceptual change. Taking these studies into consideration, it is worth highlighting that teachers in their interviews
emphasised that students they considered had passed the reported thresholds experienced conceptual changes while they valued the importance of programming in real life. This transformation the students experienced is related to the value they assigned to the course; therefore, instilling task value to students in liminal space may help them to engage with the conceptual changes that are part of the liminal journey.

Considering the first research question of this part of the study referring to task value, the findings suggest that task value did not significantly differ between the two groups since both groups experienced high levels of task value. This was interpreted as a promising finding for the liminal group since studies suggest that students with high task value engage more with conceptual change. Finally, regarding the troublesome knowledge and its impact on students’ task value, the results show that troublesome knowledge did not affect students’ task value since no significant differences were depicted between students who did the test after the questionnaire and students who did the test before the questionnaire. This strongly suggests that troublesome knowledge does not have a significant impact on the value students assign to a course, whether students are in liminal or post-liminal space. Just to note here, this study’s sample was comprised of students who personally selected to enrol in the programming course for the A/AS-level exams.

10.4 Sense of belonging, motivation, and identity in liminal space

The third point of interest in this research phase investigates students’ sense of belonging, motivation, and identity in computer science when they are in liminal and post-liminal space. Particularly, this section presents and discusses the levels of students’ sense of belonging,
motivation, and identity for both groups and also explores how these are modified when students encounter troublesome knowledge.

The results presented in chapter 8 demonstrate that generally, students in liminal space experience lower levels of sense of belonging, motivation, and computer science identity than students who are in the post-liminal space. Particularly, these differences were more intense for the computer science identity variable, and the difference is statistically significant, suggesting that students in liminal spaces demonstrate significantly lower levels of computer science identity than their peers in the post-liminal group. For the other variables—sense of belonging and motivation—even though there were differences in the mean scores between these two groups, the data failed to show that these differences were significant. In other words, students in liminal space in programming experienced fewer levels of sense of belonging and motivation in the computer science class than their peers in the post-liminal space, but not significantly lower.

Regarding the identity construct, the findings are significant. As was reported in the literature, according to Meyer and Land (2003), the threshold concepts entail a transformation of learners’ personal identity, which involves a change in values, beliefs, emotions, or attitudes. Particularly, the reconstitutive characteristic of threshold concepts refers to the change of the learner’s identity and subjectivity once the threshold concept is understood. Taking this into consideration, a difference in students’ identity between the two groups was expected. It is therefore clear from the results that passing a threshold concept involves a change in the way students see themselves in the discipline and in their role in the computing class. This finding also corroborates some of the teachers’ perspectives mentioned in the previous chapter. Teachers stated that once students cross these thresholds,
there is a boost in their ego, and students’ understandings and knowledge are enhanced and integrated, while their way of thinking reflects that of the practitioners in the field.

This finding reveals another significant point: students’ identity in computer science class may be influenced by their efficacy beliefs in the programming part of the computer science course. In other words, liminality and the uncertainty it causes to students may lead them to transfer their beliefs about their competence in programming with how they envision themselves inside the whole discipline. This is something that was evident only for students in liminal space and not for the students in the post-liminal group, and it is of great importance as it may explain the reason why students in secondary schools do not select computer science courses or drop out from them in university settings.

Regarding the two other constructs, sense of belonging and motivation (intrinsic task value) in computer science, no significant differences were evident between the two groups. This suggests that these two constructs are not affected significantly by whether students are in liminal or post-liminal space in programming. However, these results must be interpreted with caution. I remind at this point that these constructs do not measure students’ sense of belonging and motivation in programming but in the computer science course.

Many researchers have highlighted that sense of belonging significantly affects motivational constructs. According to Mucchielli (1980), membership extends to personal identification as well as a social identity, while Hagerty et al. (1996) refer to the sense of belonging as a need that drives people to construct social connections and integrate and connect with the group's members (Mucchielli, 1980; Hagerty et al., 1996, cited to St-Amand et al., 2017). Therefore, sense of belonging is strongly connected with one’s class peers and teachers. In fact, the study of Zumbrunn et al. (2014) suggests that sense of belonging is correlated with
students’ perceptions of instructor academic and social support. Additionally, their study demonstrated that students with high levels of belongness rated their instructors as “prepared, professional, respectful, more enthusiastic, passionate and caring” (Zumbrunn et al. 2014:677). Regarding the motivation (intrinsic task value) variable, many studies endeavour to explore the factors that influence students’ motivation for learning. Among them, Yilmaz et al. (2017:112) conducted a literature survey where they categorised these factors into five major themes: “teacher classroom management skills, teacher teaching methods, parent communication, student features, and school”, with teachers’ themes being the most important. Considering these, the fact that the study revealed a non-significant difference between students’ sense of belonging and motivation in computer science between the two groups might be attributed to their teachers’ instructional practices and support and their peer’s support in the computer science course. Therefore, liminality in programming did not affect students’ sense of belonging and motivation in computer science.

Trying to better understand these results, the correlation table presented in the results chapter provides some insights. In this table, it is evident that sense of belonging forms strong correlations with motivation (intrinsic task value), and self-efficacy in both groups. This finding aligns with relevant literature in this area. For instance, Zumbrunn et al. (2014) highlight that two of the motivational constructs that are correlated with students’ sense of belonging are task value (no distinction between intrinsic and extrinsic) and self-efficacy. In their paper, they discuss the direction of this relationship citing researchers (e.g. Roeser et al., 1996; Freeman et al., 2007; Ahmed et al. 2010; Midgley et al., 1989 cited in Zumbrunn et al., 2014) who have demonstrated that sense of belonging precedes the feeling of efficacy.
and task value and/or that students’ insights of their teachers’ support predict task value. Their results also point out that students who felt accepted in their class demonstrated higher self-efficacy and task value than their peers with less feelings of belonging. In other words, these studies suggest that belongingness heads students’ perception of efficacy and task value and that enhancing self-efficacy beliefs and task value may not be an easy task in an environment that does not foster belongingness (Zumbrunn et al., 2014).

The results of the current study align with Zumbrunn et al.’s (2014) as sense of belonging was correlated with self-efficacy and motivation (intrinsic task value). This demonstrates that both groups’ sense of belongingness is highly associated with the students’ intrinsic value of the course as well as their efficacy beliefs in programming. In fact, this correlation is much stronger in the liminal group than the post-liminal group indicating that students in the liminal group very closely related their feelings of belonging in the computer science class with their interests (motivation) as well as their efficacy beliefs in programming and the other way around. This last point may suggest that because students experience this uncertain and troublesome state, they tightly connect their efficacy beliefs with their feelings of belongingness and this may also have an impact on their motivation levels or vice versa.

To examine how students’ levels of sense of belonging, motivation, and computer science identity are affected by troublesome knowledge in programming, the scores of students that did the test before the questionnaire and students that did the test after the questionnaire for each group were compared. The results presented in the previous chapter show that students who responded to the affective questionnaire after the test demonstrated lower levels of these three variables, irrespective of the group to which they belonged. Moreover, when these differences were tested for statistical significance, only the sense of belonging and the
motivation variable were found to be statistically different in the two subgroups and for the liminal group. This implies that when students encounter troublesome knowledge, their motivation as well as their feelings of belonging experience a drop and are negatively affected. For the post-liminal group, these differences are restricted only in the sense of belonging variable, suggesting that troublesome knowledge has an impact on the sense of belonging even though the students have successfully passed the thresholds.

Therefore, it seems that troublesome knowledge in one part of the computer science course (programming) affects students’ sense of belonging regardless of the group they belong to while it affects motivation only for students’ in liminal space. Although the identity variable experiences a drop, the difference within the groups is not significant. While there are studies investigating the impact of students’ sense of belonging on students’ engagement and achievement, research on if and how a test influences the sense of belonging is hard to find. Thus, this research adds to this limited researched area and suggests that it is not only the teachers and peers that influence students’ class of belongingness but also students’ encounter with a performance test based on troublesome aspects of the curriculum. In fact, students’ experience with the programming test affected not only students’ self-efficacy (as it was discussed earlier) but also students’ feelings of belongingness in the computer science class without distinguishing between the liminal and post-liminal group. It seems that students may start questioning their capabilities and how well they fit in the computing classroom as a result of their encounter with troublesome knowledge. Moreover, to students that travel in the uncertainty (liminality), their experience with the programming test seems to have an impact on their motivation (intrinsic value) as well, and this may have a subsequent effect on their future engagements in the classroom.
This suggests that troublesome knowledge is an aspect, at least in the computer programming course, that it is underestimated, and more research is needed both on the learning aspects that it influences as well as on pedagogical practices that can render its learning consequences. Teachers must be very careful when designing assessment tests and should include interventions after the tests that will boost students’ levels of the examined variables. The following figure depicts the findings of this research phase. The variables in the purple (or filled) box are the affective constructs that demonstrate a significant difference between students in liminal state and post-liminal state (self-efficacy in programming and computer science identity) and within these groups (liminal group: self-efficacy in programming, sense of belonging and motivation in computer science; post-liminal group: self-efficacy in programming, sense of belonging in computer science).

![Figure 14 Variables being affected through and within liminality](image-url)
Considering the first question of this research phase referring to the constructs of sense of belonging, motivation and identity, the findings suggest that only the identity variable differs significantly between the two groups. In other words, students that have passed the corresponding thresholds experience a significantly higher level of computer science identity than do students in the liminal space. This is something that was expected, as the literature specifically highlights that once students cross a threshold, there is a shift in their identity and how they see themselves in the discipline. The other two constructs, sense of belonging and motivation, did not show any significant differences between the two groups, and this was allocated to the fact that both these constructs are highly dependent on the teacher and his/her practices and since students had the same teachers, it would have been difficult to elicit any significant differences.

Considering troublesome knowledge and its impact on sense of belonging, motivation and identity, the results indicate that the two constructs that are significantly impacted from troublesome knowledge are sense of belonging and motivation for the liminal group, and the sense of belonging only for the post-liminal group.

10.5 Regression model

The third research question of the third phase was about identifying a model that could predict students’ liminal state. To this end, two scenarios of interest were investigated and presented in Chapter 8. The first one referred to the non-reverse group - the group that responded first to the questionnaire and then to the programming test – and the other to the reverse group.
Both models can be used by teachers that are interested in determining whether students are in liminal or non-liminal space in a specific part of the curriculum with identified thresholds. From the two predictive models that were tested, one for the reverse and one for the non-reverse group, it seems that administering the test after the affective questionnaire is more efficient for predicting students’ liminal state. The corresponding model was able to correctly categorise 90% of the cases, which means that out of 40 cases, 36 were correctly identified as being in the liminal or post-liminal state by taking into consideration students’ self-evaluations, their motivation, self-efficacy and identity. The reverse case, in which the affective questionnaire was given after the test, was able to correctly predict 79.5% of the cases, which means that of the 83 participants, 65 were correctly categorised by taking into consideration all the examined affective variables with the self-evaluation significantly contributing to the model. This difference between the two models may be attributed to the troublesome knowledge and its impact on students’ affective constructs.

In the results section, another model was examined that takes into consideration all the examined variables apart from self-evaluation. This suggests that the model can predict if a student is in liminal space but without focusing on a particular part of the curriculum since the self-evaluation variable is not present. The corresponding model was able to correctly categorise 82.5% of the cases, which means that out of 40 cases, 33 were correctly identified as being in the liminal or post-liminal state by taking into consideration students’ sense of belonging, self-efficacy, motivation and identity.

Taking everything into consideration, from a methodological point of view, this part of the research suggests that affective questionnaires are best administered before performance tests if researchers are interested in depicting students’ affective domain more objectively.
Whatever the case might be, these models offer new insights into the threshold concept framework. First, the models demonstrate that it is possible to predict students’ liminal state by taking into consideration affective factors. Second, they suggest that it is also possible to determine students’ liminal state with quantitative measures, and finally, they attest that the affective domain is strongly connected with students’ liminal state and it should be acknowledged and taken into deep consideration when researchers propose pedagogical practices to help students pass through liminality more easily.

10.6 Misconception in functions

The final question of the third phase of the research focused on students’ misconceptions. The research question that this phase aimed to answer concerns with the misconceptions that students demonstrate in functions and whether or not there is a link between those misconceptions and threshold concepts. To this end, the same test that was used to categorise students in the two groups was also employed in the identification of students’ misconceptions in functions.

To identify students’ misconceptions, several methodologies have been proposed. Kaczmarczyk et al. (2010) used semi-structured interviews with a modified think-aloud protocol in which students discussed a subset of Java problems, and the analysis of students’ transcripts was based on grounded theory and qualitative analysis. Based on this work, a qualitative analysis was also used by Veerasamy et al. (2016) to reveal misconceptions in introductory programming in Python. Fleury (2000) used interviews with students and a collection of Java programs, while Sirkiä (2012) studied students’ errors in Python by examining the log files from students’ solutions to visual program simulation exercises. The
approach taken in this thesis for the identification of misconceptions in functions includes a specific part of the computing curriculum along with an assessment tool based on Bloom’s and SOLO taxonomies. Considering the results presented in Chapter 9, this specific approach seems effective for capturing the complexity of a programming area and depicting students’ misunderstandings at multiple levels.

Table 10 Misconceptions in functions

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Some students think that parameters must be explicitly defined to be used inside the function</td>
</tr>
<tr>
<td>2</td>
<td>Some students think that parameters and arguments should have the same name</td>
</tr>
<tr>
<td>3</td>
<td>Some students think that return statement should be used after the call to the function to return and print the function’s output</td>
</tr>
<tr>
<td>4</td>
<td>Some students think that parameters should have the return variable</td>
</tr>
<tr>
<td>5</td>
<td>Some students think that arguments should have the return variable</td>
</tr>
<tr>
<td>6</td>
<td>Some students think that return value must have the same name with the function</td>
</tr>
<tr>
<td>7</td>
<td>Some students think that local variables can be accessed from a function to another</td>
</tr>
</tbody>
</table>

Students who belonged in the liminal group demonstrated misconceptions while students in the post-liminal group, even though they made some errors, they did not show evidence of misconceptions in functions. In total, the study revealed seven misconceptions that the students demonstrated in functions in computer programming. The first misconception that was made by a large number of students (48.19%) points to a very specific error: Students put the return statement under the calling statement in order to print and return the result of the function. It seems that students thought that putting the calling statement first and then entering the return statement was the way to return and print the result of a function. Another error that students made quite often in the tasks—and this relates again to the return value—
is that students named the function and the return variable with the same identifier. Although it is not clear if this error happened by mistake, the fact that 24% of students demonstrated this in their code led me to believe that some students erroneously believed that the name of the variable to be returned should be the same as the name of the function. Finally, the results suggest one more misconception regarding return values: Some students put the return variable to the arguments list (misconception 5). It is likely that the students thought that in order to return a value, the variable holding this value must be placed in the set of arguments.

Trying to understand the source of these misconceptions is not an easy task. However, the literature around misconceptions in programming can offer some insights. Particularly, Perkins and Martins (1985) referred to four types of fragile knowledge: partial knowledge, inert knowledge, misplaced knowledge, and conglomerated knowledge. From these four types, it could be argued that the last two can be sources of the misconceptions described in the above paragraph. Misplaced knowledge refers to knowledge that is suitable for some cases but is employed in inappropriate cases. Conceptually, it is close enough to what Duboulay (1986) called “overgeneralization” and the garbled knowledge mentioned by Perkins and Simmons (1988), who noted the mistakes that occur when students transfer the characteristics of one command into another. Conglomerated knowledge refers to knowledge that can only roughly express intent, as is evident when students have problems following the rules underlying the way computers execute programs. This specific source of misconceptions, along with misplaced knowledge, could explain the first misconception to which the above paragraph refers. When students misplaced the return statement by putting it underneath the calling statement, it was probably because they thought that as the calling statement was part of the main function, that was the correct place for the main program to
ask the function to return the result. The same sources of fragile knowledge could explain the students’ misconception that the arguments should include the return variable. Students may think that as the communication between the function and the point that is called is through parameters-arguments, the way to communicate the result back is to put the return variable in the arguments list. Finally, students’ misconceptions about the name of the return variable may be allocated to many factors. One potential explanation is that students may have been confused about the syntax of the calling statement and regarded the name of the function as a variable that must be defined inside the function. Another potential source of confusion could be the whole conceptual idea of functions, which students may regard as a variable holding a value.

Although previous researchers like Sirkiä and Sorva (2012) have reported errors that students make with return values, this is the first time, to my knowledge, that these specific ways of handling return values are reported in the literature. A comparison of the findings with those of other studies confirms that misconceptions 1, 2, and 7 have been reported again in the literature. Specifically, misconception 1 refers to students’ erroneous understanding that parameters must be assigned a value inside the function’s body in order to be used. The same misconception was reported by Sirkiä and Sorva (2012). Additionally, misconception 7, which refers to students’ erroneous idea that local variables in the function can be accessed outside the function, was also reported by Sirkiä and Sorva (2012). Misconception 2 describes students’ misunderstandings and confusion about naming parameters and arguments. Chen et al. (2012) discussed students’ misunderstandings about formal and actual parameters and argued that these errors could be explained by a general confusion that students have with these concepts. Finally, the study found one other misconception: A few
students thought that calling a function would also print the result. Although I have included this error in the list, I am reluctant to conclude that this error stems from a misconception, as only a few students demonstrated this error.

The fact that all the above misconceptions were evident only on students in the liminal group raises a question about the relationship between threshold concepts and misconceptions. Although the relationship between misconceptions and threshold concepts is not clear in the literature, Meyer and Land (2005) regard alternative conceptions as a source of troublesome knowledge. The latter is a key characteristic of threshold concepts and, thus, misconceptions as part of troublesome knowledge may insinuate a potential threshold concept. The study here reinforces this argument, as all the concepts that caused misconceptions to students in the third phase of the research had previously been identified by the teachers (in the second phase) as threshold concepts. The question that arises then is whether all the concepts with which students experience misconceptions are also threshold concepts. From my point of view and considering that threshold concepts are more than concepts that cause difficulties to students in some part of the curriculum, I would argue that misconceptions can point to potential threshold concepts in a subject, but this does not indicate that all concepts with which students experience misconceptions are also threshold concepts. The reason is that for a concept to be considered as a threshold, it must cause a shift to the learner's view of the discipline, the way they see themselves in the discipline and perhaps the way they see the world. These changes have not yet been reported in the computer programming literature as the results of overcoming misconceptions and, therefore, I would argue that misconceptions in programming can only be seen as a starting point of investigating threshold concepts in a discipline.
Another question that also needs to be addressed is if threshold concepts cause students to experience misconceptions. By taking into consideration only my research, the results point to the argument that indeed with threshold concepts students will experience misconceptions. This was evident in this research as the concepts with which students’ experience misconceptions in functions were also found to form a threshold conception. Additionally, considering threshold concepts identified by other researchers and presented in table 2 in the literature chapter, I cannot miss noticing that most of these concepts are also concepts with which students have experienced misconceptions in programming. Thus, these observations point to the argument that threshold concepts and particularly the troublesome aspect of threshold concepts indicate that students at some point will form misconceptions or alternative conceptions.

10.7 Concluding remarks

The section presented here discusses the findings of the third phase of this thesis. This section is of paramount importance as it links the affective domain and misconceptions with the threshold concept framework. To my knowledge, this is the first study that attempts to connect these two research areas, and the findings point out that indeed, the affective domain is evident and plays an important role in liminality.

One of the most important findings of this chapter is the role of self-efficacy, identity, and troublesome knowledge. The results point out that the two constructs of self-efficacy and identity are significantly different between students in liminal and post-liminal space, with the latter students demonstrating higher levels than the former. This finding signifies that the
passage through liminality is accompanied by an increase in students’ self-efficacy in programming and computer science identity.

Additionally, troublesome knowledge plays a significant role in students’ levels of the affective variables, and surprisingly, this was evident for students in both liminal and post-liminal space. The findings suggest that students’ self-efficacy, sense of belonging, and motivation are the three affective constructs that are significantly impacted by the intersection of students with troublesome knowledge. It was, therefore, suggested that teachers should pay extra attention when they design a test and particularly should consider pedagogical practices that will boost students’ self-efficacy, sense of belonging, and motivation after a test is administered.

Another significant contribution presented in this section is that of predictive models of liminality. The models explored predicted successfully most of the students’ liminal states, taking into consideration the affective factors explored in this thesis. It was suggested that the models are stronger when the affective questionnaire is administered first and then the performance test. Overall, these models offer a new research direction in this field and a useful initial tool for teachers who are interested in investigating their students’ liminal states.

Finally, the current study set out to explore the misconceptions that key stage 5 students have with regard to functions. One of the most significant findings to emerge from this study is the identification of seven misconceptions, three of which have never before been identified in the literature to my knowledge. These misconceptions reflect the fact that the return variable seems to be confusing to students at this age and stage. The difficulties with return values apply not only to the return statement but also to the general confusion students have
regarding how functions communicate with the main program (or the point at which they are called). This finding corroborates the teachers’ reported experiences in the second part of this thesis, as all of them reported students having difficulties with these concepts. Additionally, taking into consideration that both teachers’ experiences and students’ practical reports verify the troublesome aspects of these concepts, the argument of this thesis—that the three constructs of parameters, parameter passing, and return values form a threshold conception in computer programming—is further supported.
Chapter 11 – Conclusion

Having presented and discussed the results of each phase of the study, I will synthesise the findings to form the main arguments of this thesis and provide the pedagogical implications that stem from this exploration. The following section discusses the main findings of this thesis, while the next one discusses the methodological contributions, contributions to the threshold concept framework and to the computer science education. The chapter concludes with the limitations of the study, by providing a critical reflection on the threshold concept framework as it was adapted in this thesis and by discussing future research in this area.

11.1 The thesis and main findings

The thesis attempted to explore a complicated phenomenon, that of threshold concepts, in a specific part of the computer programming curriculum focusing on functions. By adopting a mixed research design, the study managed to capture the complexity of threshold concepts both by considering teachers’ experiences teaching these concepts and students’ affective domain and experiences with troublesome knowledge. The study was not only concerned with identifying threshold concepts in computer programming. Of particular importance was investigating the transformation that students experience by considering teachers’ practice, as well as exploring the role of the affective domain in liminality by considering students’ affective dimensions of learning.

Meyer and Land (2003) posit that threshold concepts are of vital importance in curriculum design. They locate threshold concepts at the top of the design of effective learning environments, as they indicate sources of difficulties for students when grasping a concept and why a student can continually stumble while searching for a way out of a problem. From
Akerlind’s et al.’s (2010) point of view, threshold concepts play a diagnostic role for teachers as they can highlight parts of the curriculum that need special consideration. This argument demonstrates the significance of this research thesis, as identifying threshold concepts and suggesting ways for teachers to help their students as they progress through liminality will have a major impact on students’ learning experiences in computer programming.

However, as it has been addressed in the literature, identifying threshold concepts is not an easy task. Oscillating between the different researchers’ suggestions, I decided that the best approach to identify threshold concepts in functions for students at key stage 4 and 5 was to ask for experienced teachers’ opinions. In addition to this, I also determined that it would be better to employ a consensus technique, specifically the Delphi method. Therefore, the Delphi method sets the opening of this exploration, and the findings of this first part of the research were used as inputs for the second phase of the research, which is a more detailed exploration into the threshold concept phenomenon.

As I mentioned above, the first part of the research was the pillar of the whole thesis. Having the expert teachers’ consensus on concepts that are potentially threshold concepts in functions was extremely important for setting the rest of the investigation. While teachers agreed on 11 concepts that may be threshold concepts in functions, the second phase of the research shed more light on the concepts that demonstrate the characteristics of threshold concepts that were of interest in this thesis: transformative and integrative. Therefore, the second phase, which was a deeper investigation into the teachers’ experiences teaching these concepts by way of an interpretative phenomenological analysis, focused specifically on four of these concepts.
The second phase was extremely important for getting the perspectives and experiences of teachers and delving deeper into the characteristics of the corresponding concepts that may constitute them as thresholds. The teachers involved in this phase highlighted both the difficulties their students face when trying to understand these concepts as well as their experiences and perspectives regarding the transformations their students experience once they grasp these concepts and how these are portrayed. Among the changes teachers experienced with their students, of significant importance are both the advanced way of thinking and skills that students start demonstrating after understanding how parameters, parameter passing, return values and decomposition work, but also, the value that students allocate to decomposing a problem to subproblems and create functions to handle these which corresponds to the way experts in the discipline view, understand and use functions. Demonstrating skills and competencies of experts, thinking in a way similar to that of experts in a discipline, and viewing and understanding something through the lens of the discipline are the most important transformations that students experience when crossing liminality.

Therefore, the second phase shed lights to students’ experiences through their teachers’ perspectives. The results of this phase suggest that parameters, parameter passing, return values, and procedural decomposition are concepts that are troublesome for students (according to the teachers’ perspectives) with procedural decomposition being regarded as a procedural threshold or a threshold skill. Moreover, I have argued that the group of parameters, parameter passing and return values create a threshold conception in computer programming, which probably corresponds to the conception of functions. In the literature chapter, I have made clear the distinction between concepts and conceptions and how conceptions are formed by the integration of multiple concepts along with viewing them.
through the lens of the discipline. Burch et al. (2015) posit that this integration of concepts together with the appreciation of the way they are seen in the discipline generates a “lasting conception” (p.482) that gives students the opportunity to link old knowledge - in this case parameters, parameter passing and return values - in such a way that generates new knowledge - in this case functions. What this observation suggests is that if a student fails to comprehend the concepts of parameter, parameter passing and return value or either of these concepts, or if a student does not integrate these concepts in the right way or misses the view of these concepts through the lens of the discipline, they may form limited conceptions or misconceptions as argued by Burch et al. (2015). This last aspect of misconceptions as being part of the threshold concept framework was something that was examined in this research and corroborates Burch et al.’s arguments.

Kiley and Wisker (2009) advocate that once understood, threshold concepts lead to a qualitatively different way of seeing a discipline and the learning experience. These qualitatively different ways stem from the transformation the students experience when passing the threshold, which may result in the acquisition of “new knowledge and subsequently a new status and identity within the community of practice” (Eckerdal et al., 2007:124). Capturing the qualitative ways in which students experience threshold concepts is a challenging task. In the literature chapter, I have specifically highlighted the different opinions of researchers upon this issue, with some researchers arguing for qualitative interviews with teachers and lecturers and others for qualitative interviews with students. The approach in this thesis followed interviews with experienced teachers, as explained in the above paragraphs, along with quantitative measurements that were administered to students in the third phase (first part) of the study. The reason for employing a quantitative
method for the third phase (first part) of the study was that from my personal experience as a former computer science teacher, I believe that it would have been difficult for students at the age of fifteen or sixteen to effectively capture and explain the qualitatively different ways of seeing the discipline after they have grasped the threshold concepts. Therefore, as the second phase of the research was able to capture the qualitative differences the students experienced by taking into account the experiences of their teachers when teaching these concepts, the third phase aims at investigating how students experience liminality and at exploring misconceptions that students have in functions when they are in liminal space (second part).

Meyer and Land (2003) highlight that once threshold concepts are understood, a potential effect is the transformation of learners’ personal identities, which involves a change in values, beliefs, emotions, or attitudes. For Rattray (2016) as well, threshold transformations result in ontological changes which reflect both cognitive and affective changes. For this reason, the study particularly focused on exploring how students experience liminality by taking into consideration the following affective variables: self-efficacy, task value, motivation, sense of belonging, and computer science identity.

Most of the aforementioned aspects of liminality and passing the threshold were evident in this research study. Apart from students’ performance, which is naturally increased when a threshold is crossed, students demonstrate other differences when affect is considered. One such difference refers to students’ computer science identity, which was significantly more enhanced for the post-liminal group than the liminal group. This finding confirms the relationship between passing a threshold and the identity transformation that Meyer and Land (2005) emphasise. As was discussed in the literature chapter, identity is understood as
endorsed through an individual’s involvement in activities and discourses and constructed through perceptions of oneself and the participation of others as well (Carlone et al., 2014). Particularly in computer science, where the typical “geek” stereotypes portray computer scientists as smart but also lonely, antisocial, and male, it can be extremely difficult, especially for girls, to form a computer scientist identity. Although this research does not focus on handling these stereotypes or enhancing students’ identity by doing so, it does highlight the individual’s involvement in computer science activities and the importance of passing the thresholds for enhancing students’ identity in the computer science classroom.

Another important finding of this part of the research refers to students’ self-efficacy in programming. Particularly, the study revealed that students in the post-liminal group demonstrated significantly higher levels of self-efficacy in computer programming than students in liminal space. Indeed, liminality is not an easy passage as it requires time and includes feelings of nervousness, eagerness, and frustration (Eckerdal et al., 2007). In liminal spaces, the students oscillate between the known and the unknown while dealing with uncertainty (Meyer & Land, 2005). It is, therefore, reasonable to argue that students’ beliefs regarding their capacity to effectively carry out tasks will be negatively impacted when they are in liminal spaces, whereas once they cross through this uncertain stage, their efficacy beliefs will be significantly enhanced. Beliefs of self-efficacy can impact individuals’ actions and tasks, their level of effort, and their determination to pursue and continue with activities and tasks regardless of the difficulties, the amount of stress they experience, their expectations about the outcomes, and the self-regulation process. Land et al. (2014) point out that going through liminality involves the learner in a process of identifying the deficiencies in her existing understanding of the thematic area under investigation and
ultimately derogating from the older predominant view. In other words, it requires a self-regulation process. Zimmerman (2000) advocates the role of self-efficacy on students’ self-regulated learning. He argues that self-efficacious students stimulate their learning by using self-regulatory strategies and processes such as “goal-setting, self-monitoring, self-evaluation, and strategy use” (Zimmerman, 2002:87). Therefore, this study suggests that self-efficacy is a key that could help students cross through liminality. As research evidence supports, students with high self-efficacy work harder, have more persistence, are better regulated while demonstrate fewer emotional responses than students with low self-efficacy (Zimmerman, 2000). These all are important competencies necessary to pass liminality.

This study also highlighted the important role of troublesome knowledge. David Perkins (1999) refers to troublesome knowledge as something that seems counter-intuitive, alien, or incoherent. Threshold concepts can be puzzling and difficult to understand and consequently can be troublesome for students who are engaged with these concepts. In fact, the study revealed that the troublesome characteristic of threshold concepts impacts students’ affect regardless of whether students are in liminal or post-liminal space. To be more specific, the results of the study revealed that students’ levels of self-efficacy, sense of belonging, and motivation (only for the liminal group) were significantly impacted from the encounter of students with troublesome knowledge. The finding signifies that troublesome knowledge affects students’ affective domain, and therefore, teachers should be vigilant when designing test materials by employing practices that would enhance students’ affective dimensions of learning after the administration of a test.

Apart from the connection between troublesome knowledge and affect, the study also revealed a connection between troublesome knowledge and misconceptions that students
hold regarding functions. All the students that belonged to the liminal group showed evidence of misconceptions in functions which were listed in the results section. Perkins (2007) identified five types of troublesome knowledge that can be challenging for students. Among them, alien knowledge refers to knowledge that is counter-intuitive to learners’ own perspectives, and conceptually difficult knowledge refers to “misimpressions from everyday experience and reasonable but mistaken expectations” (Savin-Baden, 2007:48). In other words, both these definitions reflect misconceptions that students may hold through their experiences and perspectives interacting with the physical world. In the discussion section, I tried to elucidate the possible relationship between misconceptions and threshold concepts reaching the argument that misconceptions may lead to the identification of threshold concepts in the discipline but not all concepts with which students experience misconceptions are also threshold concepts. The results on students’ misconceptions revealed seven misconceptions in functions held by students in liminal space. It is noteworthy to mention here that three of these misconceptions refer to return values, while two relate to parameters and parameter passing, which enhances the thesis’ argument that these concepts may form a threshold conception in computer programming.

Therefore, taking everything into consideration, the study showed that liminal space provides the context for a constant battle in which students negotiate their identities, question their efficacy beliefs, and encounter different forms of troublesome knowledge, among which are misconceptions. Additionally, post-liminal students’ self-efficacy and sense of belonging were impacted after their experiences with the programming test, and this may point to the argument that post-liminal space is not a definite state and students may re-enter
liminality when they experience a stressful situation (this is discussed further in the critical reflection section).

11.2 Research contribution

In the introduction of this thesis, it was argued that mastering functions and decomposition in computer programming engage learners with different forms of troublesome knowledge and ontological and epistemological shifts during which a number of affective constructs are impacted, questioned and negotiated. The research findings presented in the above sections corroborate and reinforce the thesis statement and highlight new avenues both for the threshold concept research and computer programming education. In detail, the contributions of this thesis focus on three directions: methodological insights, contributions on the threshold concept framework, and contributions to the field of computer programming education. The following sections present the main contributions of this study.

11.2.1 Methodological insights

The following points summarise the contributions of the thesis regarding methodological aspects for identifying threshold concepts and uncovering misconceptions.

The thesis offers new methodological insights as it is the first study that

- employs a mixed approach comprising of the Delphi method along with IPA to identify and explore threshold concepts
- explores threshold concepts in secondary settings in computer programming
- uncovers specific misconceptions in functions with programming tests based on Bloom’s and SOLO taxonomy
Most studies so far adopted a qualitative approach for investigating threshold concepts with insufficient results and also focused on undergraduate students. The study presented in this thesis is the first that focuses on secondary school students in computer programming and suggests a mixed research design to identify and explore threshold concepts in functions by employing first a consensus technique (the Delphi method) and then enriching the findings with in-depth qualitative interviews. I argue that the exploration of threshold concepts is a complicated phenomenon that cannot be tackled by employing only quantitative or qualitative tools. It calls for a mixed research design with a focus on reaching consensus at the beginning of the investigation. The reason for this is that in programming, at least, every concept may be a potential threshold for some students, and this makes the identification of threshold concepts a very hard endeavour. Having a list of concepts that are potential thresholds, the researchers should engage in deep qualitative interviews or approaches (e.g. reflective essays) with students who can demonstrate experiences relevant to the characteristics of threshold concepts.

In this research, however, students may not have been mature enough to be engaged in such methods. Therefore, the interviews and the Delphi study focused on “expert” computing teachers who had many years of teaching experience as well as experience practising programming outside school settings. Reflecting on this decision, it still seems as the most appropriate way to investigate threshold concepts in secondary-school settings. Nevertheless, research that is conducted in undergraduate settings should focus both on first- and second-hand experience by including students as well as academics. Finally, the study also identified students’ misconceptions by employing a programming test based on Bloom’s and SOLO taxonomy. By employing programming tasks of different and increased difficulty
and specifically focused on a particular part of the programming curriculum, the study was able to uncover students’ misconceptions in functions, three of which have never before been reported in the literature. Reflecting on this methodological choice, it seems as an effective approach for uncovering misconceptions and extends the functionality of the above taxonomies.

**11.2.2 Threshold Concepts Framework**

The following points summarise the contributions of the thesis regarding the threshold concept framework.

The current thesis contributes to the threshold concept framework as it is the first study that

- empirically explores the affective domain in the threshold concept framework and therefore, in computer programming
- considers the relationship between misconceptions and threshold concepts empirically
- suggests an initial affective quantitative model for identifying liminal state
- suggests that post-liminal space may not be definite, and students may revisit liminality

Regarding the threshold concept framework, the thesis has offered new insights regarding the way students experience liminality and, therefore, extends the framework of threshold concepts. In contrast to other research conducted in this area, the study here is the first that empirically links the affective domain with the threshold concept framework. The study identifies possible threshold concepts in computer programming and considers how students experience the state of liminality regarding the affective domain, the role of troublesome
knowledge on students’ affective domain, the connection between misconceptions and threshold concepts and additional methods for identifying students’ liminal state. The results of this investigation provide empirical evidence that corroborates Meyer’s and Land’s arguments regarding students’ efficacy beliefs and identity shifts once thresholds are crossed and further elaborates and highlights the role of troublesome knowledge on students’ affect. It was argued in this thesis that students’ efficacy beliefs and computer science identity are significantly impacted by liminality, whereas students who successfully pass the thresholds demonstrate significantly higher levels of these two variables.

Another interesting finding refers to the role of troublesome knowledge in students’ affective domain. The results of the study demonstrated that troublesome knowledge is not something that only affects students in the liminal space but also students in the post-liminal space. Although this study does not offer a clear explanation for the impact of troublesome knowledge on students’ affect, some initial suggestions and thoughts were highlighted. Learning how to program and overcoming learning difficulties it is a challenging process and therefore, it is possible that students’ previous experiences in programming and the difficulties they had encountered to influence their affective state. other words, while students in the liminal space confront troublesome knowledge in order to reach a more precise conceptualization of a phenomenon, or because they try to integrate existing knowledge with new, students in post-liminal space need to leave behind the stressful experiences they underwent trying to pass through liminality and enhance their efficacy beliefs. Another possible explanation may concern the thin lines between the liminal and post-liminal state and therefore, the possibility the students that were categorised in the post-liminal group not to have completely left the liminal space (this is discussed more on the
critical reflection section). Taking these into consideration, the study suggests a model that uses students’ self-efficacy, motivation, identity, and self-evaluations as indicators for predicting the liminal state. The thesis also attempted to explain the relationship between misconceptions and threshold concepts, pointing out that misconceptions are part of the troublesome aspect of threshold concepts, and thus, can suggest possible threshold concepts in a discipline.

11.2.3 Computer programming education

The following points summarise the contributions of the thesis regarding computer programming education and are discussed in detail in the paragraphs that follow.

The current thesis contributes to computer programming education as it is the first study that

• suggests that in programming, threshold conceptions are central to students’ understanding and can be thresholds to students’ progress
• suggests that in the area of functions, procedural decomposition is a procedural threshold
• suggests that in the area of functions, parameters, parameter passing and return values form a threshold conception
• suggests three new misconceptions in functions regarding return values

Regarding the contributions in the computer programming education field, the current thesis is the first that considers the role of (threshold) conceptions in computer programming and distinguishes them from (threshold) concepts. Specifically, by interviewing expert teachers in this field, it was clear that students in computer programming have difficulties understanding single concepts but also struggle with connecting and integrating new
knowledge to form conceptions. For example, “functions” should not be treated as a single concept, but as a conception that incorporates other concepts like parameters, parameter passing, and return values each of which poses a conceptual burden in understanding functions and how these actually work. Therefore, the study is the first to suggest that parameter passing, parameters and return values form a threshold conception in programming and that procedural decomposition is a procedural threshold. Additionally, the study contributes to the computer programming education field by suggesting three new misconceptions in functions referring to the same concepts.

Regarding the contributions of this thesis in respect to the way teachers should approach teaching functions, these refer mostly on building a safe affective environment for students to experience liminality. Therefore, although I could not offer direct guidelines to teachers on how they should approach teaching these concepts (this would require further research and intervention designs), indirect suggestions are stemming from the research findings.

From this study, it was evident that the affective domain and specific misconceptions in functions are evident when students are in liminal space. As the study revealed, students can face many problems, both conceptual problems as well as those stemming from the transformative and integrative aspects of threshold concepts. This suggests that teachers, apart from considering only the misconceptions or other sources of troublesome knowledge that their students face, could also consider students who struggle integrating new knowledge with the old and support them through the transformations that this process entails.

In line with this, this research study sheds light on how strong this process can be and the potential impact it may have on students’ emotional capital. It is important for teachers to acknowledge that going through a threshold influences the learner’s self-efficacy and
identity, as was evident in this study. Teachers should accompany their students to this journey, ease their discomfort and help them boost their efficacy beliefs, enhance their identities in respect with the subject and create supportive environments in which students feel safe to experience these transformations.

This study also considered three constructs that are particularly important for students’ engagement: motivation, sense of belonging, and task value. I have highlighted in the discussion section the important role of teachers and their practices in enhancing students’ sense of belonging and motivation. Students can remain engaged as long as they feel members of their class and continue to value the importance of the course for their personal/professional development. Therefore, teachers should help students feel part of the computing class, keep their motivation high, and highlight the importance of the course for their students’ personal and professional goals. Additionally, teachers should not only incorporate opportunities to enhance students’ self-efficacy but also incorporate self-evaluation tasks while monitoring them for any inconsistencies. In this way, students will engage with self-assessment opportunities that could potentially help them evaluate their performance and capabilities more objectively, take control of their learning, and become self-regulated learners.

In this research, it was also evident from the teachers’ experiences that students, when they learn the concepts of parameters, parameter passing and return values, revisit other concepts that they have met before (e.g. variables, control flow). The challenge then for teachers is how to design courses that do not follow a hierarchical approach but an approach that would allow students to go back and forth to their understandings and knowledge.
Therefore, seeing the concepts of parameters, parameter passing and return values as a threshold conception and procedural decomposition as a procedural threshold, teachers should create a supportive learning environment which constantly helps students to enhance their efficacy beliefs and identity in the computer science class, feel members of the class, appreciate and value the importance of their course to their personal growth and/or professional development, confront their misconceptions or alternative conceptions and reach a precise conceptualisation, self-regulate their learning, and revisit their understandings, inform them or transform them accordingly. The latter point is of particular interest as generally the subject of computer programming is regarded as sequential and teaching with a spiral curriculum approach has not been studied much.

11.3 Limitations

The limitations of the study centre mostly around the sample selection and the methodologies employed. In the following paragraphs, I describe in detail the limitations of each phase.

In the first phase, the limitations involve the sample, as only 10 teachers participated in the Delphi study. The Delphi method sets 10 participants as a down limit and focuses more on the expertise of these participants rather than their actual number. From this perspective, the size number was acceptable, although including more participants may have provided some more insightful results. Although I am not that concerned about the size of the sample, I find that the different perspectives of the participants regarding the threshold concept framework pose a stronger limitation to the study. To elucidate, this limitation refers to the participants’ perspective regarding what a threshold concept is, what characterises a threshold concept and what is the difference between a difficult concept, a misconception and a threshold
concept. While before the first round of the Delphi I sent to the participants articles that could help them understand the perspective with which the threshold concept framework is presented by Meyer and Land and also develop their own understandings of this framework, I am reluctant to believe that all the participants engaged with this process. Therefore, it is possible that the participants in the first phase of the study based their responses on an arbitrary understanding of what a threshold concept might be. Finally, another limitation of this phase is that the sample consisted of experienced teachers with a subset having practised programming outside school settings. Deciding who would be most suitable to identify threshold concepts is still a methodological challenge and it was discussed in the literature chapter. Having explored only teachers’ opinions in this study, I would argue that future studies should consider a more diverse sample. Ideally, research studies should consider professionals programmers who demonstrate an excellent understanding of how concepts are linked together and who can share and explain how understanding a concept led them to a different way of viewing programming or a part of it. Furthermore, academics with focus on programming and education studies should also be considered and offer important insights in this research area.

For the second phase, the limitations include the sample but not the size of the sample. In an interpretative phenomenological analysis, the number of participants may be as low as one or two, as the aim is for the researcher to depict the individual experience of the participants (idiography) as well as interpret the whole account of their experiences. Thus, the limitation in this phase is not concerned with the number of the sample but with the individuals who took part in the study and their capacity to provide meaningful experiences with respect to the phenomenon under study. In the second phase, computing teachers formed the sample
and provided their experience teaching the corresponding concepts, what they perceive as difficult or troublesome for their students, and what changes they have observed on their students. Therefore, this phase provides second-hand experience, as it investigates teachers’ perspectives about students’ experiences. I understand that asking teachers to reflect on their practices and understandings about how their students experience a phenomenon is difficult and may be judged as an inaccurate way of approaching threshold concepts. However, as I have already mentioned in the literature and methodology chapters, asking secondary students about the ontological and epistemological changes that they experience once specific concepts were understood may have been an extremely difficult and unreliable endeavour; therefore, my attention in this phase of the research has focused on teachers whose experiences teaching these concepts can give us awareness of students’ experiences. It is the teacher’s job, through teaching, to observe and interpret and try to understand the difficulties that students encounter in the course as well as to understand when the students finally surpass their problems by observing changes in their attitudes, emotions, behaviours and performance. As Allison and Pissanos (1994:47) argue, “Observing hold a key position in the cycle (observing, interpreting, decision making) because interpretation of classroom events and, consequently, pedagogical decisions are dependent on the observational abilities of the teacher”.

Additionally, IPA is not only a method for investigating the first-hand experience. Pietkiewicz and Smith (2012:362) argue that “The primary goal of IPA researchers is to investigate, how individuals make sense of their experiences. It is assumed that people are as self-interpreting beings (Taylor, 1985), which means that they are actively engaged in
interpreting the events, objects, and people in their lives”. Therefore, IPA does not exclude the interpretation of an individual’s experience about another individual in their life.

Finally, in the third phase, the limitations refer mostly to the test applied, the way that the students were separated in the two categories and the sample. The test given to the students was created only by me, and therefore, it included what I considered relevant regarding the threshold concepts uncovered in the previous phase. To address this problem, and as I mentioned in the Methodology section, the test was created according to Whalley et al.’s (2016) guidelines and examples of tasks that belong to each of the Bloom and SOLO taxonomies. Additionally, two researchers categorised the students into the liminal or post-liminal group, and interrater reliability was examined in the Results section of this thesis.

Another problem may stem from the way the students were separated into the two groups. I need to reiterate here that students were separated into the liminal and post-liminal group by following specific criteria (SOLO taxonomy) mentioned in the Methodology section. These criteria are based on the assumption that the group of parameters, parameter passing and return values are threshold concepts and as such, should always be interpreted under this assumption. In other words, the SOLO taxonomy should be used to uncover students’ liminal state only if the tasks are designed in this way as to address threshold concepts uncovered previously.

Another limitation stems from the participants. The sample consisted of A-level students who voluntarily selected to attend computing courses in their school. This may have an impact on the affective factors examined in this thesis, as the task value that these students allocate to computing courses was high. Therefore, the results presented in this thesis should always be interpreted with this limitation in mind. Additionally, because only 123 students
took part in this thesis, the results could not be generalised. On top of that, the nature of threshold concepts posits that students may experience their journey differently through liminal space, and, therefore, this thesis attempted to describe a possible trend among students who experience liminality and students who have passed the thresholds and not to generalise the findings to the whole population.

11.4 Critical reflection

Looking critically at this thesis, many questions have arisen that I had not anticipated or thought when I first designed the research. When I started this exploration, I had a very strong belief in the existence of threshold concepts in every discipline and particularly in programming. Although that belief hasn’t changed, questions have arisen regarding the practical side of threshold concepts, the liminal and the post-liminal state in secondary-school settings and whether the focus of research in this framework and particularly in computer programming should be only on single concepts or a group of concepts.

To elucidate, at the onset of this research, I believed that threshold concepts would be experienced in a similar way among students. After these three years of research in this area, I realised that if threshold concepts exist in a discipline, then these may be different for each student, and each student may experience these differently. A concept that may be captured quite easily from one student may be a struggle for another, and the reasons for the difficulties may be different among students. Particularly in computer programming courses, I believe that every concept could be a threshold for some individuals, and if we could interview many individuals, we would end up with a threshold list with all concepts of programming in it. One reason for this is that in programming, we have underestimated the
role of conceptions when these are seen as a group of concepts. For instance, when Eckedräl and her colleagues argue that object-oriented programming is a threshold concept, I would say that a more accurate characterisation would be that of a threshold conception with concepts like objects and classes being part of this conception and being the troublesome aspects of object-oriented programming. Therefore, instead of only searching for single concepts that could be thresholds (which was mostly how misconceptions were investigated), we should turn our focus to the strong and troublesome connections that concepts in programming form and explore how we could bring about these connections to our teaching. It is only then when students’ transformative learning can start taking place and when they can start appreciating the subject. I need to note at this point that I am not arguing that there are no single concepts that are thresholds in programming, although I believe that these are few (e.g. variables); I am arguing for a turn in our focus as researchers and teachers to more complex ideas in programming that cannot be expressed by a single concept and whose troublesome and transformative characteristics emerge when other interconnected concepts come into view.

However, uncovering all these connections between concepts is an extremely difficult endeavour in programming, and it is even more difficult bringing them in secondary settings. I doubt that students in secondary education would ever reach such a deep and multi-connected understanding, which I think it can only be accomplished by years of practice. In other words, reaching a post-liminal space as it is conceptualised by the threshold concept framework may be extremely difficult in secondary-school settings for two reasons. Firstly, the post-liminal state requires many years of practice and exposure to learning experiences that will strengthen one’s understandings, ideas, views about the discipline and views about
oneself. Thus, the question that I raise here is a question of expertise: Is it possible for such a deep appreciation of a subject to be reached before students enter universities and start practising programming in a more professional way? This is extremely difficult, and if it can be achieved in secondary-school settings, it would only be by very few students. Secondly, learning is a continuous process, and as students and more generally, learners gain experiences, their understandings, conceptualisations and views are transformed. Taking myself as an example, when I graduated from university, I thought that I had an excellent understanding of programming; however, I came to realise aspects, concepts and ideas of programming in a different and more complex way when I started teaching the subject. Does this indicate that I had never reached the post-liminal state as a student? Students and learners may reach a state where they feel confident with their understandings, conceptualisations of the subject and identities within the discipline, but this may not be a state that lasts forever. When new experiences are coming into view and make learners question what they know or revisit their understandings, transformations occur, and that is something that is happening throughout life. Thus, the question that arises here concerns the stability of the post-liminal state: Is it possible that the post-liminal state is not an absolute state, but liminality is revisited as learning progresses?

Reflecting on the current research findings, I understand that students who were part of the post-liminal group had resolved misconceptions and other difficulties they may have experienced. They have overcome the difficulties stemming from troublesome knowledge, whether these focused on misconceptions or misunderstandings or skills that needed to be obtained or difficulties in viewing connections between their previous knowledge and the new one. They had a secure understanding of what they have learned so far, and that may
have given them the confidence and strong computer science identity in this discipline. However, considering the impact that the programming test had on their efficacy and sense of belonging, which was not expected for students who have passed the thresholds, strengthens the argument that post-liminal space may not be a permanent state and as soon as these students advance their knowledge in programming or another discipline and gain new experiences, their previous knowledge as well as their identities will be questioned, and it is that moment when they may enter liminality again. This realisation does not contradict Meyer’s and Land’s argument that threshold concepts are irreversible (once understood they cannot be forgotten). Rather, it expands this characteristic of threshold concepts by supporting that a threshold concept’s understanding cannot be forgotten but it can be enhanced. The conceptualisation of liminality as a recursive process mirrors Bruner's (1960) spiral curriculum where ideas and concepts are initially introduced and mastered in a simpler or basic level and further revisited and reconstrued in a higher level. The following diagram depicts my perspective on a spiral approach to encountering a threshold concept.
Figure 15 Reflecting on Liminal state with a spiral perspective

Therefore, while I am not questioning that these two groups – liminal group and post-liminal group - had differences (quantitative and qualitative) and were correctly separated into two groups, I am questioning the stability of a post-liminal state, especially for secondary-school students.

Finally, one more critical point refers to implications on teachers’ practice. Even if Meyer and Land have suggested some teaching considerations for threshold concepts (presented in the literature chapter), these are general and should be part of every teaching pedagogy not only those targeting threshold concepts. That made it very difficult for me to suggest practical suggestions to teachers on how they should approach pedagogically this part of the curriculum, although some points were mentioned in the previous sections. I believe that the framework of threshold concepts needs further considerations and research under this specific area.
Concluding this section, I believe that this thesis offers new insights into the threshold concept framework as well as new considerations for the computer programming education community. In the first chapter of this thesis, I highlighted the importance of this investigation in secondary-school settings where programming has been embedded in the school curriculum. After this investigation, I further consider that it is important, when threshold concepts are considered in secondary-school settings, the focus of teachers and further research to turn to two major goals: first, on assisting students to overcome problems stemming from the troublesome and integrated aspects of thresholds and secondly, on equipping students with the necessary cognitive and non-cognitive skills and practices that will help them be lifelong learners, persist through difficulties and engage in the variety of transformations that learning and new knowledge entails.

11.5 Future research directions

The research presented in this thesis offered some new insights into the threshold concept framework and in functions in programming. Nevertheless, the learning and teaching of programming would benefit a lot if further research could be conducted in this area with a particular focus on the following research directions.

First of all, one future research direction would be to uncover more threshold conceptions and procedural thresholds in the discipline of programming. As I mentioned in the previous section, I strongly believe that as students progress in the discipline, they encounter more difficulties regarding the way concepts are related and connected to each other. The transformations related to threshold concepts seem more reasonable to be occurred in programming by a group of concepts rather than a single concept. Therefore, future research
should focus on exploring these threshold conceptions in programming and help students to understand the inter-connectedness of the concepts included. For example, it would be worthwhile exploring further the conceptual understanding of variables and parameters as perceived by the students and how this is changed as students progress in the discipline. The teachers highlighted that once students grasp the concept of parameters then the concept of a variable is extended. In general, however, parameters and variables are the same thing and what may be confusing is the reason for being used. It is possible, therefore, that the first threshold concept that students encounter in programming is the concept of variables, which is partially understood until students encounter functions in programming. From that point, they need to accommodate a new, extended meaning and use of the previously formed understanding of variables bringing them again to a liminal state. The discussion presented in section 11.4 about liminal space being perceived as a spiral process reinforces this argument. A study that will track students’ understandings about variables and how these are transformed as students progress in the discipline by highlighting the “critical events” that led to these transformations could shed more lights to this research area.

A second aspect worth investigating refers to students’ transformations. The research here focused on these transformations but from the teachers’ perspectives because I judged that students in upper secondary classes would not be able to describe these changes. However, transferring this research to undergraduate settings would shed light on the transformative experiences students undergo to pass through liminal space. Ideally, such research should be conducted both by employing quantitative tools, such as the ones presented here, supplemented by deep qualitative interviews and reflective practices (e.g. reflective journals). Additionally, one of my future goals would be to follow university students from
the first year of their studies until the last one, and record their experiences, how the
transformations came about and what triggered these changes. In this research endeavour, it
would be worth considering other sources (apart from students) that could contribute in the
identification process. For instance, academics who have extensive experience teaching
programming in undergraduate settings could significantly contribute to this process. Their
experience could point to troublesome aspects of programming and places in the
programming curriculum that the majority of undergraduate students get stuck. Additionally,
since this research would refer to undergraduate students it would be of substantial
importance to explore how threshold concepts transform students’ professional identity and
their views of the discipline. To this end, professional programmers could also be included
in the research, not only to reflect on troublesome aspects of programming concepts but also
to shed lights on concepts or conceptions that helped them build their professional identity
and shape the way they view the discipline. Professional programmers could also contribute
on identifying threshold conceptions - how concepts in programming form a concept web
with the relations between concepts being the threshold to students’ understanding (the
integrative aspect of threshold concepts).

A third point of consideration would be to explore gender differences, particularly focusing
on the sources of troublesome knowledge and the strategies female and male students use to
exit the liminal space. Additionally, in this investigation, of particular importance would be
the role of the affective domain and the potential differences female and male students
experience relating to the affective constructs. This idea was initially part of this thesis, but
because the female participants were few, no further statistical analysis was conducted.
Finally, one area that lacks research regards practical pedagogical implications for teaching the subject of programming. While lots of research has been conducted regarding the identification of misconceptions and thresholds, there are few research papers referring to how actually teachers should confront these. Therefore, research should focus on school interventions that would practically explore and evaluate strategies to confront misconceptions and thresholds in this discipline and produce guidelines for teachers that would help them build transformative environments and for students to integrate their knowledge and experience these transformations as safely as possible. An initial starting point for me would be building and testing a spiral curriculum for programming that would allow students to revisit previous knowledge and transform it while integrating a new one. Additionally, these learning environments should consider students’ emotional capital by acknowledging that affective constructs such as self-efficacy and identity are questioned through liminality and students’ motivation, and sense of belonging should be retained at high levels, especially after students’ knowledge is evaluated.

Considering both the results of this study as well as the above research directions, it is obvious that research in this area of programming is still new and ongoing. Lots of research work needs to be done towards building a computer programming curriculum that recognises the multi-facet difficulties that students experience and assist them towards building confidence, competence and potential expertise in this subject. To this end, the computer programming subject needs more intervention studies that would evaluate empirically the impact of these interventions on students’ learning and that would take into consideration both students’ and teachers’ experiences, perspectives and criticism towards building a
strong programming curriculum focusing on helping students acquire ways of thinking apparent in the world they live in.
12 Bibliography


Boo, H.K. (2005). Teachers' misconceptions of biological science concepts as revealed in science examination papers. At the Australian Association for Research in Education Conference, Parramatta, 27 November - 1 December 2005, Australia


Internet WWW page at URL: http://www.it.uu.se/edu/course/homepage/datadidaktik/ht10/reports/Self-Efficacy.pdf (accessed at 06/10/2017)


Hancock, B. (1998) Trent Focus for Research and Development in Primary Health Care: An Introduction to Qualitative Research. England, Trent Focus


Mezirow, J. (2000). Learning as Transformation: Critical Perspectives on a Theory in


ACM.


Plass-Oude Bos, D., (2015). Identifying and addressing common programming misconceptions with variables (part 1), Master's thesis, University of Twente


Royal Society (Great Britain), (2012). *Shut down or restart?: The way forward for computing in UK schools*. Royal Society.


Smith, J.A. (2004). Reflecting on the development of interpretative phenomenological analysis and its contribution to qualitative research in psychology. *Qualitative Research in Psychology*, 1, 39-54


13 Appendix

13.1 Consent form for teachers

CONSENT FORM FOR PARTICIPANTS IN RESEARCH STUDIES

Please complete this form after you have read the Information Sheet and/or listened to an explanation about the research.

Title of Study: Identifying and Addressing threshold concepts in computer programming at KS4 and KS5 by exploring students’ and experts’ thinking processes

King’s College Research Ethics Committee Ref: / MR/16/17-295/

Thank you for considering taking part in this research. The person organising the research must explain the project to you before you agree to take part. If you have any questions arising from the Information Sheet or explanation already given to you, please ask the researcher before you decide whether to join in. You will be given a copy of this Consent Form to keep and refer to at any time.

I confirm that I understand that by ticking/initalling each box I am consenting to this element of the study. I understand that it will be assumed that unticked/initialled boxes mean that I DO NOT consent to that part of the study. I understand that by not giving consent for any one element I may be deemed ineligible for the study.
1. I confirm that I have read and understood the information sheet dated 18th May 2016 for the above study. I have had the opportunity to consider the information and asked questions which have been answered satisfactorily.

2. I understand that my participation is voluntary and that I am free to withdraw at any time without giving any reason. Furthermore, I understand that I will be able to withdraw my data up to 2 weeks after my interview.

3. I consent to the processing of my personal information for the purposes explained to me. I understand that such information will be handled in accordance with the terms of the UK Data Protection Act 1998.

4. I understand that my information may be subject to review by responsible individuals from the College for monitoring and audit purposes.

5. I understand that confidentiality and anonymity will be maintained, and it will not be possible to identify me in any publications.

6. I consent to my interview being audio recorded.

7. I guarantee to maintain the confidentiality of the focus group discussion.

__________________  __________________  _________________
Name of Participant    Date               Signature

__________________  __________________  _________________
Name of Researcher     Date               Signature

13.2 Information sheet: Phase 1

INFORMATION SHEET FOR PARTICIPANTS

REC Reference Number: [MR/16/17-295]
You will be given a copy of this information sheet

Identifying and Addressing threshold concepts in computer programming at KS4 and KS5 by exploring students’ and experts’ thinking processes

I would like to invite you to participate in this research project which forms part of my PhD research. You should only participate if you want to; choosing not to take part will not disadvantage you in any way. Before you decide whether you want to take part, it is important for you to understand why the research is being done and what your participation will involve. Please take time to read the following information carefully.

The study

The aim of the study is to identify threshold concepts in computer programming at KS4 and KS5 and to explore how students can be helped in understanding these concepts. For this reason, I am inviting both teachers of computer science who have experience in teaching computer programming at KS4 and KS5 (group1) and teachers who have industrial experience (practising programming) and experience teaching at any stage (group2).

If you decide to take part in group1, you will contribute to the identification of threshold concepts by a three round process (30-40 minutes per week/for three weeks) where you will be asked to give your professional opinion in a relevant question. The communication will be held by e-mail with the researcher only. Additionally, we are asking you to participate in face to face interviews about your experiences on teaching the concepts identified (50 minutes). The interviews will be recorded.

If you decide to take part in group2, you will contribute to the identification of threshold concepts by a three round process (30-40 minutes per week/for three weeks) where you will be asked to give your professional opinion in a relevant question. The communication will be held by e-mail with the researcher only. Additionally, we are asking you to participate in face to face interviews about your expertise on approaching a computer programming task designed for students (60 minutes). This process will require you to think aloud and it will be recorded.

We would appreciate your help with this. Your participation is voluntary and you are free to withdraw at any time. All data will be anonymised immediately after collection and will be stored securely. Audio recordings will be destroyed once the interviews have been transcribed.
I will produce a final report summarising the main findings, which will be sent to you. I also plan to disseminate the research findings through publication and conferences within the UK.

If you have any questions or require more information about this study, please contact me using the following contact details:

Maria Kallia
PhD student
School of Education, Communication and Society
King’s College London
Email: maria.kallia@kcl.ac.uk

If this study has harmed you in any way or if you wish to make a complaint about the conduct of the study you can contact King’s College London using the details below for further advice and information:

Dr Sue Sentance
Senior Lecturer in Computer Science Education
School of Education, Communication and Society
Kings College London
Franklin-Wilkins Building WBW 1/11
150 Stamford Street, London SE1 9NH
sue.sentance@kcl.ac.uk

Thank you for reading this information sheet and for considering taking part in this research.
13.3 Questionnaire phase 1: Delphi study

Delphi Process
Round 1

Leading Researchers
Maria Kallia, PhD student: maria.kallia@kcl.ac.uk
Dr Sue sentence: sue.sentance@kcl.ac.uk

Anticipating return date: 23/03/2017

Background Information
These questions are about you, your education and the time you have spent in teaching. In responding to the questions, please mark the appropriate box.

1. What is your gender?
   - Female
   - Male

2. What is your latest level of education?
   - Bachelor's degree
   - Master's degree
   - Doctoral degree
   - Other

3. What is your current job?
   - Teacher
   - Work in industry
   - Lecturer
   - Post-Doc

4. How many years have you worked as a teacher at ks4 and/or 5?
   - None
   - 1-3 years
   - 4-6 years
   - 7 or more years

5. Do you have experience working in industry?
   - No
   - 1-3 years
   - 4-6 years
   - 7 or more years

**Information**

6. Do you think that more research is needed in how teachers should approach teaching core concepts or constructs in computer programming at ks4 and ks5?
   - No
   - Yes

7. Were you familiar with the framework of threshold concepts?

8. Do you believe that the framework of threshold concepts can explain some of the students’ difficulties in computer programming?

9. Would you change the way you approach a concept or a construct if you knew that this was a threshold concept?

10. Do you believe that students encounter more difficulties in understanding theoretically certain concepts or constructs?

11. Do you think that students encounter more difficulties in applying their knowledge in the programming tasks?

12. Do you think that students can effectively write code even though they haven’t really grasped the theory behind the concepts or constructs they employ?
13. Do you think that students have difficulties in applying their understanding in programming tasks even though they have a theoretical understanding of the corresponding concepts and constructs?

**Task Description**

Threshold concepts are concepts that students get stuck and are unable to progress; once understood though, they transform the way students see their disciplines or subjects. A threshold concept may be defined by a single word such as “objects” but may be something defined by a phrase rather than few words.

Students face many difficulties in computer programming. For this study, we turn our focus in functions and procedural abstraction. This is an area that teachers and researchers have identified as challenging and difficult for students.

**Based on your experience, can you please suggest two or more threshold concepts on the area of functions and procedural abstraction?**

Please provide a short description of each of the suggested threshold concepts and be aware that a threshold concept does not necessarily has all the five characteristics, but it is essential to have the integrative and transformative. You do not need to explain in this stage why the concept you suggested has the above characteristics. We just need a short description in case another participant suggests the same concept with you, but his/her articulation is different.

13.4 **Questionnaire phase 2: IPA study**

Basic interview questions
1. From your experience, what are the difficulties students face with parameters?
2. From your experience, what are the difficulties students face with parameter passing?
3. From your experience, what are the difficulties students face with return values?
4. From your experience, what are the difficulties students face with procedural decomposition?
5. Have you experienced any changes that happen to students once they grasped the concept of parameter passing? For example, is there another concept or something else that students understand in programming as soon as they grasp parameter passing?

6. Have you experienced any changes that happen to students once they grasped the concept of return values?

7. Have you experienced any changes that happen to students once they grasped the concept of parameters?

8. Have you experienced any changes that happen to students once they grasped the concept of procedural decomposition?

9. Based on your teaching experience, do you think that students have more difficulties in understanding the concept of procedural decomposition or applying it in programming exercises?

13.5 Information sheet phase 3

INFORMATION SHEET FOR TEACHERS

REC Reference Number: LRS-17/18-4282

YOU WILL BE GIVEN A COPY OF THIS INFORMATION SHEET

Identifying and Addressing threshold concepts in computer programming at key stage 4 and key stage 5
I would like to invite you and your students from your school to participate in this research project which forms part of my PhD research. This project is being carried out by researchers at King’s College London. The students should only participate if they want to; choosing not to take part will not disadvantage them in anyway. Please take time to read the following information carefully.

The study

Computer programming is a complex cognitive ability that requires a variety of skills. The conceptual understanding of this subject contributes to its difficulty and threshold concepts, as a source of troublesome knowledge, have a significant role in this. Thus, identifying threshold concepts and developing effective learning and teaching methods for programming are increasingly important. This research project will contribute to the teaching framework of computer programming at schools. The research findings would lead to greater understanding of the difficulties students are facing in computer programming. Then, possible suggestions would be made on how teachers might rethink and potentially restructure the curriculum more effectively to reduce students’ conceptual difficulties and provide a more dependable, comprehensible and positive learning environment.

You have been given this information sheet about our project as we would like you to inform the students of your school (Year 11-GCSE computer science, Year 12 and 13-AS/A level computer science) about our study. They will be asked to fill out a questionnaire related to their self-efficacy, motivation and identity (7 points Likert scale) in the computer programming class and to solve some computer programming exercises in Python. This process will take 30-40 minutes, and it could be held in lunchtime or after-school session or during the computer science course. The questionnaire and the tasks will be completed and submitted in an online platform. The link will be provided to you in order to guide students appropriately.

Once your students have taken part in the online questionnaire, we will analyse the data for our research project. We can also provide you with some information about your class and how they are performing in Python and how this relates to their motivation and self-efficacy. This will be useful for tailoring your future teaching.

Students under the age of sixteen will need their parents’ consent to participate. Older students do not need their parents’ consent.

Participation is voluntary, and if the students choose not to participate, this will have no impact on them. The students are also free to withdraw at any time without giving any reason, but this would
not be possible after the submission of the responses. All data will be anonymised and will be stored securely.

We would appreciate your help with this.

If you have any questions or require more information about this study, please contact me using the following contact details:

Maria Kallia

PhD student

School of Education, Communication and Society

King’s College London

Email: maria.kallia@kcl.ac.uk

If this study has harmed you in any way or if you wish to make a complaint about the conduct of the study you can contact King’s College London using the details below for further advice and information:

Dr Sue Sentance

Senior Lecturer in Computer Science Education

School of Education, Communication and Society

Kings College London

Franklin-Wilkins Building WBW 1/11

150 Stamford Street, London SE1 9NH

sue.sentance@kcl.ac.uk

Thank you for reading this information sheet.

13.6 Questionnaire and programming test phase 3

Questionnaire for students ks4/5
Your instructor is participating in a study of school teaching and learning, in cooperation with the University of King’s College London. We would like to ask for your participation in the study. As part of the study, you will be asked to fill out a questionnaire related to your motivation and learning in the computer programming class and to solve some programming tasks. Your participation is voluntary and not related in any way to your grade in this class. You may decide to participate now but you can withdraw from the study at any time. All your responses are strictly confidential and only members of the research team will analyse your responses.

<table>
<thead>
<tr>
<th>Participation code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
</tr>
<tr>
<td>Year 11, 12, 13</td>
</tr>
</tbody>
</table>

**Part 1**

These questions ask you about your attitudes and value beliefs about your GCSE/AS/A level Computer science course, particularly with reference to learning programming. There are no right or wrong answers. We would like you to respond to the questionnaire as accurately as possible, reflecting your own attitudes and behaviours in this subject.

If you think the statement is very true of you, circle 7; if a statement is not at all true of you, circle 1. If the statement is more or less true of you, find the number between 1 and 7 that best describes you.
<table>
<thead>
<tr>
<th></th>
<th>Not at all of me</th>
<th></th>
<th>Very true of me</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I believe I will receive an excellent grade in the programming part of this course</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. I am certain I can understand the most difficult material presented on the readings in programming</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. I am confident I can understand the basic concepts taught in programming.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. I am confident I can understand the most complex material presented by the teacher in programming.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. I am confident I can do an excellent job on the assignments and tests in programming.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. I expect to do well in the programming part of this course.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. I am certain I can master the skills being taught in programming.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Considering the difficulty of programming, the teacher, and my skills, I think I will do well.</td>
<td>1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In the following questions, if you think the statement is very true of you, circle 6; if a statement is not at all true of you, circle 1. If the statement is more or less true of you, find the number between 1 and 6 that best describes you.

<table>
<thead>
<tr>
<th>Question</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>9. I think I will be able to use what I learn in programming in other courses</td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>10. It is important for me to learn the course material in programming.</td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>11. I am very interested in the content area of programming</td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>12. I think the programming course material is useful for me to learn</td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>13. I like the subject matter of this programming</td>
<td>1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>14. Understanding the subject matter of programming is very important to me</td>
<td>1 2 3 4 5 6 7</td>
</tr>
</tbody>
</table>

I feel that I am supported in this computer science course | 1 2 3 4 5 6 |
I feel that I am a part of this computer science course     | 1 2 3 4 5 6 |
I feel that I am accepted in this computer science course    | 1 2 3 4 5 6 |
I feel comfortable in this computer science course           | 1 2 3 4 5 6 |
I like my computer science course                            | 1 2 3 4 5 6 |
I think my computer science course is interesting            | 1 2 3 4 5 6 |
I enjoy my time in computer science course                   | 1 2 3 4 5 6 |
I like doing the activities we do in my computer science course | 1 2 3 4 5 6 |
Part 2: Programming tasks

Task 1

The following program has 7 lines of code. Some parts are underlined and annotated with a letter. Match each of the following concepts with the appropriate letter in the code.

<table>
<thead>
<tr>
<th>Concept</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>i. parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ii. arguments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iii. calling a function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iv. return statement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v. function definition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vi. print statement</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. `def preference(breakfast, lunch):`
2. `print("I prefer ", breakfast, ", lunch, ", for breakfast and ", lunch, ", for lunch.")`
3. `dinner = "salad"
4. `return dinner`
5. `def main():`
6. `e_meal = preference("toast", "salad")`
7. `print("I prefer ", e_meal, ", for dinner")`

Task 2

The following program has 7 lines of codes. However, there are some errors. Identify these errors and correct them providing your answer in the text box below. There is only one error in each line or no error at all. If there is not an error in a line, write “No error” in the corresponding field below.

1. `def calculate age (year):`
2. students_age = 2018 - year
3.
4. def main()
5. yearofbirth=int(input("type the year of birth:"))
6. age = calculate_age()
7. print ("the student's age is " age)

<table>
<thead>
<tr>
<th>Number of Line</th>
<th>Error description</th>
<th>Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Task 3

The following program has 19 lines of code. Read the program carefully and try to answer the following questions:

Program A

1. def decision (type):
2. if type = "expensive":
3. print ("unable to buy!")
4. else:
5. print ("you can buy the product!")
6.
7. def price_evaluation (price):
8. if price < 10:
9. eval = "cheap"
10. elif price < 20:
11. eval = "normal"
12. else:
13. eval = "expensive"
14. return eval
15.
16. def main()
17. cost=int(input("type the product’s price:"))
18. evaluation = price_evaluation (cost)
19. decision (evaluation)
Questions

1. What is the purpose of this program? (understanding-multistructural)

_________________________________________________________________
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________

2. if the value of the cost variable is 50: (application-multistructural)
   a. what would the price_evaluation function return?
_________________________________________________________________
   b. What would the outcome of this program be?
_________________________________________________________________

Task 4
The purpose of the following program is to calculate and print the result of a test (pass or fail). There are two functions: the main(), which reads the mark on a test and prints the results and the output(), which calculates if the student has passed or failed the test.

The code statements are syntactically correct but are misplaced. Can you put the statements in the right for the program to achieve its goal?

1. def output (grade):
2. def main ():
3. result = "pass"
4. if grade > 12:
5. mark = 13
6. print output(mark)
7. else:
8. result = "fail"
9. return result
Indicate the right order of the above lines of code:

Task 5

The following program takes as an input two numbers and calculates their sum by calling a function. Fill in the blanks in order for the program to achieve its purpose.

```python
def sum(____A______):
    ___B___ = n1 + n2
    return result

def main():
    number1 = int(input("type the first number"))
    number2 = int(input("type the second number"))
    print sum(____C____)
```

Task 6

Write a program that calculates and prints the area of a rectangle. The program should have:

a. A function named `area` that calculates and returns the area of a rectangle
b. A main function that takes as an input the length and the width of a rectangle and prints the rectangle’s area (you should call the function `area` for this purpose)
Please answer the last two questions.

<table>
<thead>
<tr>
<th></th>
<th>Very bad</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>How do you think you did in the test?</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>How do you think you get on with the computer science course?</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

13.6.1 Tasks and Bloom and Solo level

<table>
<thead>
<tr>
<th>Question</th>
<th>Type of task</th>
<th>Bloom</th>
<th>SOLO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic</td>
<td>Remember</td>
<td>Unistructural</td>
</tr>
<tr>
<td>2</td>
<td>Errors</td>
<td>Remember/Understand</td>
<td>Unistructural/Multistructural</td>
</tr>
<tr>
<td>3</td>
<td>Code intent</td>
<td>Understand</td>
<td>Relational</td>
</tr>
<tr>
<td>3</td>
<td>Tracing</td>
<td>Apply</td>
<td>Multistructural</td>
</tr>
<tr>
<td>4</td>
<td>Parson puzzle</td>
<td>Apply</td>
<td>Multistructural</td>
</tr>
<tr>
<td>5</td>
<td>Skeleton code(Gaps)</td>
<td>Analyse</td>
<td>Relational</td>
</tr>
<tr>
<td>6</td>
<td>Write a program</td>
<td>Create</td>
<td>Relational</td>
</tr>
</tbody>
</table>
28 February 2021

TO: Maria Kallia

SUBJECT: Confirmation of Registration for “Identifying and addressing threshold concepts in Computer Programming for 14-17 years old students by exploring experts' and students' thinking processes”

Dear Maria

Thank you for submitting your Research Ethics Minimal Risk Registration Form. This letter acknowledges the receipt of your registration; your Research Ethics Number is MR/16/17-295. You may begin collecting data immediately.

Be sure to keep a record your registration number and include it in any materials associated with this research. Registration is valid for one year from today’s date. Please note it is the responsibility of the researcher to ensure that any other permissions or approvals (i.e. R&D, gatekeepers, etc.) relevant to their research are in place, prior to conducting the research.

Record Keeping:

In addition, you are expected to keep records of your process of informed consent and the dates and relevant details of research covered by this application. For example, depending on the type of research that you are doing, you might keep:

- A record of the relevant details for public talks that you attend, the websites that visit, the interviews that you conduct
- The ‘script’ that you use to inform possible participants about what your research involves. This may include written information sheets, or the generic information you include in the emails you write to possible participants, or what you say to people when you approach
them on the street for a survey, or the introductory material stated at the top of your on-line survey.

- Where appropriate, records of consent, e.g. copies of signed consent forms or emails where participants agree to be interviewed.

**Audit:**

You may be selected for an audit, to see how researchers are implementing this process. If audited, you will be expected to explain how your research abides by the general principles of ethical research. In particular, you will be expected to provide a general summary of your review of the possible risks involved in your research, as well as to provide basic research records (as above in Record Keeping) and to describe the process by which participants agreed to participate in your research.

Remember that if you have any questions about the ethical conduct of your research at any point, you should contact your supervisor, the Research Ethics office, or a member of your Department’s Research Ethics Panel for advice.

**Feedback:**

If you wish to provide any feedback on the process you may do so by emailing crec-minrisk@kcl.ac.uk.

We wish you every success with this work.

With best wishes

Research Ethics Office
13.8 Ethics approval – phase 3

Mara Kallis

27 November 2017

Dear Mara

LRS-17/18-4282

I am pleased to inform you that full approval for your project has been granted by the E&M Research Ethics Panel

- Ethical approval is granted for a period of three years from 27 November 2017
- You should report any untoward events or unforeseen ethical problems to the panel Chair, via the Research Ethics Office, within a week of occurrence. Information about the panel may be accessed at: http://www.kcl.ac.uk/innovation/research/support/ethics/committees/osh/index.aspx
- If you wish to change your project or request an extension of approval, please complete and submit a Modification Request to proc-lowrisk@kcl.ac.uk. Please quote your ethics reference number, found at the top of this letter, in all correspondence with the Research Ethics Office. Details of how to complete a modification request can be found at: http://www.kcl.ac.uk/innovation/research/support/ethics/applications/modifications.aspx
- All research should be conducted in accordance with the King’s College London Guidelines on Good Practice in Academic Research available at: http://www.kcl.ac.uk/college/policyzone/assets/files/research/good%20practice%20Sept%202009%20FINAL.pdf

Please note that we may, for auditing purposes, contact you to ascertain the status of your research.

We wish you every success with your research.

Best wishes,

Miss Annah Whyton

Senior Research Ethics Officer

For and on behalf of:
E&M Research Ethics Panel