

UNIVERSITY OF KWAZULU-NATAL

**FACTORS INFLUENCING THE LEARNING OF INTRODUCTORY
COMPUTER PROGRAMING AT THE DURBAN UNIVERSITY OF
TECHNOLOGY**

By

Kelvin Samuel Osaji-Onalo

216076915

**A dissertation submitted in fulfillment of the requirements for the
degree of**

**Master of Commerce Coursework in Information Systems &
Technology**

**School of Management, IT and Governance
College of Law and Management Studies**

Supervisor: Dr. S Ranjeeth

2019

DECLARATION

I, *Kelvin Samuel Osaji-Onalo* declare that

- (i) The research reported in this dissertation/thesis, except where otherwise indicated, and is my original research.
- (ii) This dissertation/thesis has not been submitted for any degree or examination at any other university.
- (iii) This dissertation/thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
- (iv) This dissertation/thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
 - a) Their words have been re-written but the general information attributed to them has been referenced;
 - b) Where their exact words have been used, their writing has been placed inside quotation marks, and referenced.
- (v) Where I have reproduced a publication of which I am an author, co-author or editor, I have indicated in detail which part of the publication was actually written by myself alone and have fully referenced such publications.
- (vi) This dissertation/thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation/thesis and in the References sections.

Signature:

Date: *11/09/2019*

ACKNOWLEDGMENTS

My profound gratitude goes to my Supervisor, Dr. *Sanjay Ranjeeth* for his support, guidance, advice, encouragements and constructive feedback all through the completion of this dissertation. He has made me a far better student and researcher than I was. He never accepted less than my best efforts. He was a blessing to me and also the best research supervisor any student could wish for. Many thanks to all the staff in the discipline of Information Systems and Technology who have imparted knowledge to me in one way or the other, I say a big thank you.

To my administrators, Vaneshree Govender, Prem Mohun and most especially Hazvinei Muteswa who is always there for the students. I say a big thank you to you all.

I would also like to thank my Master's program lecturers, most especially Dr. Prabhakar Rontala Subramaniam, and Dr. Given Mutinta, who not only encouraged me but also gave me valuable advice.

A very big thank you to my Family, Mrs. Marry Osaji-Onalo, my siblings and their partners, Princess Anupu and Ogene, Prince Eng. Azubike Gideon and Ngozi, I will not stop until I mention Princess Chekira and Eng. C.E Ofoluwa, Prince Obiajulu Ignatius and Roseline, Prince Boniface and Eziego, Prince Monday Metelojo, Princess Uyo Osaji-onalo, Prince Monday, Princes Chukwuma and Bose, Princess Ego Osaji-Onalo, also Princess Thereza for all their love, care and moral support, guidance and encouragements. I cannot thank you all enough; you have been very supportive all through this journey. To my nephews and niece, Monday, Regina, Luis, Ada, Bright, Esther, Onome, Ese, Tega, Edirin, Ojode, Udego, Atadele, Ndudi, and Junior, I say a big thank you and I love you all.

My sincere appreciation to my friends, Mc MacDon, Ifanyi Ego, Baby Chimamanda, Onyebuchi Celestine, Jude Ike, James Onuoha, Mthoko, Ebuka,Vusi, John, Rasta Kaka, Andrew Oranwa, Tapiwa, Pillay Rajaan, Mariah Jordan, Attwood Heidi, Obi Richard, Ugeshnie, Ravesen Moodley, Charles, Kibwe, Sinqobile Ndimande, Dlamini Wenzile and my lovely wife Osaji-Onalo Ajuma who have made this journey colourful, exciting and enjoyable. In addition, to my superior colleagues who I consulted for assistance, Prof. Seraphin Eyono Obono, Cheryl Rose White, Dr. Delysia Timm, Avenal Jane, Jairajh Roopraj, Jude Kala, Lawrence De Law, Obafemi Samson, I say a big thank you to you all.

I will also use this opportunity to remember the late Zuko Mbombo, my class mate and my best friend from under-graduate education (B-Tech) at the Durban University of Technology till the last phase of our Master's Degree level (MCom) at the University of KwaZulu-Natal. I just want to say "I love you Zuko" and I will uphold the dreams we shared together. Thank you for everything.

To my Father in the Lord Prophet T.B Joshua whom God used to break the captivity of limitation in my life. You have pointed a direction which I should go the way of Cross of Calvary. One of the quotes that keep me going, He said "Don't mistake GOD silence as rejection. His love is constant and unchanging. For, God has a time for everything, a time to be born, a time to grow, a time to face persecution, a time to overcome and a time to show the proceeds of victory i.e. time to reap the product of victory"

Finally, thanks to EmmanuelTV teams and glory to God Almighty who has made it possible for me to complete this dissertation.

LIST OF ABBREVIATIONS

APPDEV	Application Development
DUT	Durban University of Technology
ICT	Information and Communication Technology
ID	Intellectual Development
IT	Information Technology
MFILCP	Model of Factors that Influence the learning of Computer Programming
NSFAS	National Student Financial Aid Scheme
UKZN	University of KwaZulu-Natal
UOT	University of Technology

ABSTRACT

FACTORS INFLUENCING THE LEARNING OF INTRODUCTORY COMPUTER PROGRAMING AT DUT

Computer programming is an extremely difficult skill to master for students who are novice computer programmers. The preceding assertion is based on reports of high failure rates in introductory computer programming courses offered by tertiary education institutions. This is not just a South African problem but a number of cross-institutional and multi-national studies show that the problem is well known and is common (Grover et al., 2016).

The current study investigated the factors influencing the learning of introductory computer programing at Durban University of Technology (DUT). The objectives of the study were to understand the influence of previous experience on students' learning of introductory computer programming as well as to understand the influence of self-efficacy on students' learning of introductory computer programming. The study also focused on understanding the influence of the 'mental model' representation of the problem domain on students' learning of introductory computer programming, and to understand the influence of the 'mental model' representation of the problem domain on students' self-efficacy in the learning of introductory computer programming. The study adopted the quantitative research method to investigate the subject matter. This study embraced a survey research strategy and data collection carried out was over a short period. The study used simple random sampling to select 200 respondents at DUT. Data were collected using questionnaires. Data quality control was ensured by conducting a reliability and validity test on the data collection instrument used in this study. Ethical approval for the study was obtained from DUT. The quantitative

data collected were analyzed using the SPSS, version 25.0. The study utilized statistics such as frequency, descriptive (mean and standard deviation) and inferential statistics (Cronbach's alpha and Spearman correlation). The overall findings from the study suggested that the self-efficacy level of the research participants was high. The results of the study revealed that there was a moderate positive relationship between self-efficacy and computer programming. Furthermore, it found that the mental model adopted by students when solving computer programming problems positively influences student performance in computer programming.

An outcome of the study is the recommendation that the teaching and learning of computer programming should focus on language structure and the correct mental interpretation of the problem domain so that students could improve their performance.

Table of contents

<i>Supervisor's Permission to Submit</i>	<i>Error! Bookmark not defined.</i>
<i>Declaration</i>	<i>ii</i>
<i>Acknowledgments</i>	<i>iii</i>
<i>List of Abbreviations</i>	<i>v</i>
<i>Abstract</i>	<i>vi</i>
<i>List of Tables</i>	<i>xi</i>
<i>List of Figures</i>	<i>xii</i>
<i>1.0 Introduction</i>	<i>13</i>
1.1 The Challenge of Learning Computer Programming	13
1.2 Background and Context for the Study.....	14
1.3 Research Questions and Objectives of the Study.....	15
1.4 Research Rationale	17
1.5 Significance of the Study	18
1.6 Outline of the Study.....	18
1.7 Summary	19
<i>2.0 Literature Review</i>	<i>21</i>
2.1 Introduction.....	21
2.2 The Teaching of Introductory Computer Programming	21
2.3 Factors that Influence the Learning of Introductory Computer Programming	28
2.4 The Study's Theoretical Model	32
2.5 Summary	34
<i>3.0 The Research Methodology</i>	<i>36</i>
3.1 Introduction.....	36
3.2 Research Philosophy.....	38
3.3 Research Approach	39
3.4 Research Strategy	40
3.5 Research Design	41

3.6	Research Choice	42
3.7	Time Horizon	42
3.8	Research Site and Setting.....	43
3.9	Population of the Study.....	43
3.10	Sampling and Sampling Technique	44
3.11	Data Collection Instrument	45
3.12	Summary	49
4.0	Data Presentation, and Discussion of the Findings.....	50
4.1	Introduction.....	50
4.2	Reliability: Cronbach’s Alpha Coefficient	51
4.4	Overview of the Questionnaire Design	52
4.5	Descriptive Statistics.....	54
4.6	Mental Model.....	58
	<i>Descriptive statistics: Key dimensions of the study</i>	<i>62</i>
4.7	Inferential Statistics	63
4.8	Spearman Correlation	63
4.9	Chapter Summary	66
5.0	FINDINGS AND CONCLUSION.....	68
5.1	Introduction.....	68
5.2	The influence of previous experience	68
5.3	The influence of self-efficacy on students’ learning.....	70
5.4	The influence of the ‘mental model’ representation on students’ learning	71
5.5	Implications for Computer Programming Pedagogy	72
5.6	Conclusion	75

References.....	76
Appendix A: Questionnaire	84
SECTION B (Self-efficacy):	88
SECTION C (Mental model):	90
SECTION D (Suggested Enhancement):.....	91
Appendix B: Ethical Clearance.....	92
Appendix C: Research Approval Letter	93
Appendix D: Request For Permission To Conduct The Research.....	95
Appendix E: Similarity report	96
Appendix F: General factor	97
Appendix G: participants opinion of computer programming self- efficacy.....	98
Appendix H: corresponding statements used for correlation.....	99
Appendix J: Declaration Certificate Of The English Language Editing Of The Dissertation.	101

LIST OF TABLES

Table 4.1 Reliability: Cronbach's Alpha Coefficient	51
Table 4.2 Validity: Factor analysis.....	Error! Bookmark not defined.
Table 4.3 KMO and Bartlett's Test.....	52
Table 4.4: Overview of Questionnaire Design.....	53
Table 4.6 Demographic information	56
Table 4.7 The mental models identified in the study	60
Table 4.8 Descriptive statistics: Key dimensions of the study	62
Table 4.9 Mental Model and Self-efficacy	64
Table 4.10 Self-efficacy and previous experience.....	65
Table 4.11 Previous Experience and Mental Model.....	66

LIST OF FIGURES

Figure 2.1: The MFILCP adapted from Wiedenbeck <i>et al.</i> (2004).....	33
Figure 3.1 The research Onion taken from Saunders <i>et al.</i> , (2017, p. 54).....	37
Figure 4.1 Self-efficacy.....	57
Figure 4.2 Number and percentage of participants for each model of value assignment	61

1.0 INTRODUCTION

1.1 The Challenge of Learning Computer Programming

Owing to the growth in information and communication technologies (ICT), the need for graduates with a high competence in computer programming skills is ever increasing in knowledge-based economies around the globe. Likewise, market analysis has shown that the level of investment in computer technology software-related industries is rapidly growing compared to technological hardware-related industries, which is a phenomenon that puts an emphasis on the learning of programming (Chen, 2017). As a result, such opportunities pose new challenges particularly the challenges that influence students' learning of introductory computer programming.

However, despite the demands on human sources for high competence and skills in computer programming, there are plenty of studies in the literature which report that there is a deficiency in computer programming training initiatives (Groen&Hosseini, 2017). It is reported that computer programming courses at university level have seen a student dropout and failure rate as high as 30%, therefore indicating that programming is a challenging activity (Hosseini, 2017).

Whilst the afore-mentioned factors such as previous experience, self –efficacy, mental model and performance are usually beyond the control of lecturers, they are of the opinion that the learning style adopted by a student of computer programming influences that student's mastery of computer programming concepts. The main objective of this study is to examine the influence of various factors on

the learning of computer programming at a tertiary educational institution. It is envisaged that such a study would enhance pedagogical knowledge thereby contributing to an improved pass rate (Carpenter *et al.*, 2016).

1.2 Background and Context for the Study

According to Bawa (2012) the former Vice-Chancellor of DUT, higher education has emerged as a key ingredient in the development strategies of democratic countries. Bawa (2012) further maintains that there is a growth unprecedented in higher education's provision for increased participation rates to promote equity of access to the historically disenfranchised who seek to realize their potential through tertiary education. A goal of the national higher education system is to build institutions with new organizational forms, identities, and cultures as integral components of a single coordinated national higher education system (Horton & Craig, 2015). This goal seeks to transform the national higher education landscape into a constitutionally acceptable pedagogic endeavor devoid of any barriers to entry based on social or ethnic standing.

There are disparities within South Africa's schooling system, which is the feeder system into the higher education system (Balfanz *et al.*, 2012). Many of the students registering for courses such as ICT do not have prior exposure to the field from their high school training. At the Durban University of Technology (DUT), many students struggle to cope with the challenges of academic study. These students are therefore required to register for foundation courses in order to obtain the pre-requisite knowledge needed to cope with the learning environment. This challenge exacerbated by the difficulties inherent in

the learning of computer programming. There are many factors that may influence students' learning of computer programmings such as the lack of prior ICT exposure and proficiency in mathematics or science-related subjects (Schiefele, 2017). These factors were been examined as part of this study. The study conducted will be in the Department of Information and Communication Technology, within the Faculty of Accounting and Informatics at DUT. The researcher employed is currently within this Department as a lecturer in computer programming to first and second-year students.

1.3 Research Questions and Objectives of the Study

It is common among South African first-year students registered in IT/ICT Departments to find the introductory computer programming courses difficult. These difficulties are as a result of factors such as students lacking prior exposure to an ICT environment; lack of their knowledge of the basic use of computers; inadequate students' problem-solving ability which could involve mathematics competency; self-efficacy; attitude toward learning; and as well as other factors (Thomsett-Scott, 2016). According to Schoeman and Gelderblom (2016), attitude is equally as important as capability. Students admitted into higher education remain disadvantaged and underprepared. This has serious implications for the state, the economy, academic institutions and the youth at large (Cutts *et al.*, 2006). Underperformance and dropout of students from the system at increased rates is a costly issue for the University and therefore there is a major concern about the high dropout rate of students at tertiary institutions globally.

Huang and Shiau (2017) state that, improving graduate outcome in terms of numbers, quality and productivity, is crucial for South Africa's future. Thus, first-year students' adjustment and successful transition from high school to university are of great concern, both nationally and internationally. These issues, therefore, necessitate an investigation into factors influencing comprehension of and training in introductory computer programming language with a particular focus on first year ICT students at DUT.

The research questions and research objectives of this study are as follows:

1.3.1 Research Questions

- How does 'previous experience' influence students' performance in the learning of computer programming
- How does Self-efficacy influence students' performance in the learning of computer programming
- How does the 'Mental Model' representation of the problem domain influence students' performance in the learning of computer programming
- How does the 'mental model' representation of the problem domain influence students' self-efficacy in the learning of computer programming

1.3.2 Research Objectives

The primary objective of this study is to investigate the factors influencing the learning of introductory computer programming with a

particular focus on first year ICT students at DUT. The research objectives are as follows:

- To evaluate the influence of previous experience on students' learning of introductory computer programming
- To evaluate the influence of self-efficacy on students' learning of introductory computer programming
- To evaluate the influence of the 'mental model' 'representation of the problem domain on students' learning of introductory computer programming
- To evaluate the influence of the 'mental model' representation of the problem domain on students' self-efficacy in the learning of introductory computer programming

1.4 Research Rationale

The rationale behind this study was that little research was done in the past to investigate factors that influence first-year students' learning of computer programming in the African context and more specifically in the South African context (Hjorth, 2017). Again, existing research conducted has been mostly in Computer Science and Technology course-related institutions and not in a university setting which includes a University of Technology (UoT). Reports shows that among other factors the four basic influences on computer programming skill are previous experience, self-efficacy, mental model, and student's attitude (Bringula & Aviles, 2017) while some other authors have added to this list by including previous experience as the de facto requirement

for computer programming skill (Edgcomb *et al.*, 2017). These studies have shown that exposing students to prior ICT knowledge has a positive effect on student performance in Computer Programming courses at tertiary level. It is envisaged that such an intervention will adequately prepare these students for the demands at the tertiary level (Grover *et al.*, 2016).

1.5 Significance of the Study

The focus of this research is on identifying which conceptual factors lead to success in learning computer programming, and, equally, which students are most likely to face difficulties in the course due to their inability to grasp the concepts. By identifying the vulnerable population, the ICT department and lecturers can assist these students early through recommendations with the goal of helping improve performance in the learning of computer programming. As a lecturer of a computer-programming course at the DUT, the researcher has observed that many first-year students face difficulties in the course leading to a high failure and dropout rate. This study will add to existing knowledge on the factors that influence the learning of introductory computer programming and it will provide a platform for future research even in different contexts.

1.6 Outline of the Study

Chapter One introduces the study including the background of the study, the research problem, the research questions, research objectives, research rationale and the significance of the study.

Chapter Two reviews the existing literature conducted on factors influencing first-year students learning computer programming, exploring their gaps and weaknesses, hence justifying the need for this study. Basic introduction of computer programming discussed. This chapter also critically reviewed models available for selection, identifying their gaps and constructs, hence explaining and justifying the model adopted for this study, displaying the constructs in order to explain the various factors considered in this study.

Chapter Three describes the methodology adopted for this study. The research philosophy, research strategy, research choice of data collection, research design, study site, population, sample and data collection procedure of the study were all discussed. This chapter also described the data collection instruments, as well as ethical considerations for this study.

Chapter Four presents and analyses the data obtained in this study. Tables, bar graphs, and pie charts used were for better representation of the descriptive statistics. Similarly, an inferential statistical test used was to provide a detailed explanation of results and to explain relationships between variables considered in this study.

Chapter Five presents the conclusion of the study. It also confirms that the research questions addressed and that the research objectives achieved. The limitations of the study, recommendations, and suggestions for future research provided will be in this chapter.

1.7 Summary

This chapter has introduction provided to the study, the background, the research problem, the research questions, research

objectives, research rationale and the significance of the study. The structure of the dissertation presented was also in this chapter.

The following chapter provides an overview of the different types of computer programming languages; it also reviews various literary contributions to the discussion of the factors influencing ICT students' academic performance by identifying gaps in the body of knowledge. Similarly, it also critically reviews the theoretical model used in the study to explore the factors influencing previous experience and self-efficacy in the learning of computer programming.

2.0 LITERATURE REVIEW

2.1 Introduction

This chapter seeks to explain the research questions posed in the study by discussing the several factors that influence the learning of computer programming. Most importantly, the chapter will comprehensively analyze different literature on introductory computer programming and what makes students find programming skill difficult to achieve success in academic performance, exploring gaps within the literature, hence justifying the need for this study.

The model adopted for this study is been explained, illustrating the constructs in order to clarify the various factors the variables considered in this study and justifying why it was considered appropriate for this study.

2.2 The Teaching of Introductory Computer Programming

Introductory computer programming language usually taught in the tertiary institution by means of a series of lectures. Mauer *et al.* (2017)state that these lectures cover the simple concepts of programming (variables declaration, method, loops, conditionals, architectural ties and so on). These concepts are illustrated using the syntax of a particular language and more details of the language are added regularly as the students become more familiar with the course (Watson, 2013).

Computer programming involves quite a number of tasks, including planning, coding, testing, debugging, deploying and

maintaining the programs source code (Creswell & Poth, 2017). Students in tertiary institutions progress in these skills through a series of courses. According to Barnes *et al.* (2017) in introductory courses, learners learn a certain portion of each of the activities involved in computer programming. Some courses emphasize coding and debugging tasks from the beginning and some focus on design activity first (Coles and Phalp (2016). However, there is no proof that any specific approach has an effect on learners' pass rate. In programming, one needs to note the following:

- Without visualizing the principal tasks such as execution of an algorithm in pseudocode and flowchart form, writing or reading a chunk of code is impossible.
- Debugging activity cannot be successful without a clear understanding of what each line of code does and what they do collectively as a snippet.
- Designing also heavily relies on understanding the capability and limitation of programs (Rex & Roth, 1998).

Success in an introductory computer programming course demonstrates a student's ability to visualize the logical processes behind the execution of programs (Thomsett-Scott, 2016). This ability is an important foundation, which makes all programming tasks easier, and without it, learning computer programming may be difficult. The dynamics of students to which computer programming have changed over the years and a first year students' computer programming course now has to be able to meet the needs of a highly diverse set of students. Hašková *et al.* (2014) say that computing is also a degree course that is

assumed by many to lead directly to a creative and innovative career in the IT industry (Rodrigo *et al.*, 2009). It has been shown simultaneously that many students are likely to embark on the course with the view to gaining a highly paid job as the sole aim (Kumar & Laakso, 2016). These students will have little interest in computing (or programming) other than as a means to an end (Coles & Phalp, 2016).

2.2.1 Completion Rate in Introductory Programming

Although the problem of high failure rates in introductory programming has become a global phenomenon, according to Bain and Wilson (2017) few studies have focused on providing reasons for this occurrence. Chen and Cheng (2012) conducted a survey of five educational institutions teaching computer studies to first-year students. In general, they found that the pass rate was higher for smaller classes and for colleges rather than universities, but that the programming language used does not matter. Chi and Berger (2017) undertook a longitudinal study by searching articles published between 1960 and 2013 to find those that reported data on failure rates. In all, they found 54 articles that described failure rates in 161 computer-programming courses at 51 institutions across 15 countries from 1979-2013. The worldwide mean for passing was 67.7%. but no study was there a common definition of 'passing,' whether anything above an F or only those grades that allowed a student to continue to the next course and whether or not the passing rates counted course attrition as well as failure. However, the means in the two studies being as close as they were presents a good argument that the population means is around 67% (Chi & Berger, 2017).

2.2.2 Students' Success in Learning Introductory Computer According to Bowlick and Goldberg (2017) in order to improve the pass rate for introductory computer programming, there is a need to have a better idea of what factors contribute to the high percentage of failure. One possibility is that the abstract content essential for programming is too logical for some students to grasp fully. Miller and Ramirez (2017) developed a tool to predict success in an introductory computer-programming course based on Jean Piaget's intellectual development (ID) levels. The actual level is categorized by the use of logic applied to diagnose the problems. It involves inductive thinking, but not logical thinking (Thomsett-Scott, 2016). The formal level is characterized by theoretical and logical thinking and the ability to use symbols associated with abstract concepts in a logical way. Barker and Unger's instrument had 11 questions that were categorized as concrete, early formal, formal or late formal. Answering both early formal questions (direct proportion and probabilistic thinking) incorrectly placed students in the late concrete category. If either was answer correctly the student was placed in early formal, and if, in addition, the student answered three out of four of the late formal questions (propositional and correlational reasoning, deductive logic or permutations) they were categorized as late formal. Idemudia *et al.* (2016) suggest that those who struggle with abstract thinking may need more time to internalize the concepts upon which programming depends.

Breese *et al.* (2017) sought a means of filtering out students less likely to succeed in programming in response to a high demand for the course that the faculty could not meet. They found some correlation

between math and verbal statistics scores and success, but these along with the other factors they considered (rank in high school and grades in prior exposure to ICT and problem-solving skill) accounted for at most 25% of the dissimilarity of grades (Carpenter *et al.*, 2016). Elarde (2016) tested a model with 12 predictive factors, which included problem solving skill background, contributions for success/failure (explanations students give for their success or failure on the midterm exam), domain-specific self-efficacy, mental model, comfort level in the course, favorite work style, previous programming experience and previous non-programming computer experience. Comfort level, mathematics and a competitive work style preference positively correlated with performance in the midterm, while attribution of performance in the exam to luck or the difficulty of the task negatively correlated with performance in the midterm examination. Furthermore, while prior programming experience, in general, did not show any effects as they found a prior formal class in programming to be predictive of success. In addition, while other computer experience (internet, games, and office applications) in general did not have any effects, hours playing computer games did have a negative influence. Lopez and Whalley (2008), tested a different set of predictors namely spatial visualization, reasoning designing, sketching a map as well as attitudinal factors, to see which correlated with success. Some of their correlations are statistically significant, though not strong. Others are not statistically significant unless they also include the students who did not complete the course. Bringula and Aviles (2017), found a trend toward students who created survey maps that modeled both the routes and the landmarks to be stronger at programming than those who sketched out

routes or landmarks alone. They also found students who could better articulate their search strategy more efficient than those who were less articulate. In general, their analyses are analogs to programming, therefore, someone who can create a more complete abstract model or who can articulate their search methodology in more detail shows better performance in problem-solving in a domain involving abstraction and algorithmic thinking (Mathews, 2017).

2.2.3 Motivation and Habits of Students Who Failed

Barnes *et al.* (2017) looked at differences between students who were taking the course again and new students. First-year students generally take their introductory programming course, and the first semester has an enrolment of 400 students. The second semester has an enrolment of 80 students. The study conducted was in the second semester of 2016, whereof the 80 students enrolled, 58% were taking the course for at least the second time. Barnes *et al.* (2017) did this survey halfway through the semester and another at the end of the semester. They found that most of the repeat students had little interest in programming, but initially wanted to go into business school not information technology. The repeat students also worked significantly more hours at jobs outside school than the students new to the course. Many of the repeat students had poor attendance at lectures. The authors described the repeat students as having a shallow learning approach, being reluctant to seek out and explore extra resources using their own resourcefulness. Many of the repeat students did not use or own the textbook, though it was strongly recommended. While about a quarter of the new students failed the course, over a third of the repeat students failed the class again. This study suggests a lack of motivation

to learn programming is the primary concern, but since the study was done after the students had failed the course for the first time, their lack of motivation and interest in other majors courses may in some cases be a result rather than a cause of their failing the first time.

2.2.4 Performance in Introductory Programming Courses

Predicting performance in introductory programming courses is a widely studied problem, and the motivation behind these studies is usually the high failure rates. Ideally, the study wants to be able to recognize the students who are struggling early on during the course, so that those students can then be offered additional help and support (Johnson-Laird, 1983). On the other hand, successful students could offer additional challenges to improve their learning experience. Understanding the reasons behind failing or succeeding can help to plan these interventions and teaching methods in general. Carpenter *et al.* (2016) are of the opinion that factors related to students' background, such as previous academic success and previous programming experience as well as psychological and mental model factors, self-efficacy and self-esteem influence first-year students' learning of computer programming. Some studies have also included demographics like gender and age (Hjorth, 2017). More recently, as it has become more and more common to collect log data on introductory programming courses, newer studies have also included variables based on this data. These variables try to capture students' behavior while they are solving exercises, for example by taking into account how much times they spend dealing with errors.

When predicting performance, an important thing to consider is how to measure it (Bain & Wilson, 2017) said that most studies have

focused on predicting performance on introductory programming courses, the final grade or midterm grade is a natural choice as a measurement scale. In most cases, the grade consists of performance in exercises and the final exam, though final exam usually makes up for most of the grade. Exam and lab performance have also been examined separately and other more specific performance measures have been used.

2.3 Factors that Influence the Learning of Introductory Computer Programming

There are numerous social and cognitive factors that influence the learning of computer programming (Wiedenbeck *et al.*, 2004). However, for the purpose of the current study, the most significant of these factors are discussed below.

Previous Experience and Computer Programming

Research shows that previous programming experience has a positive effect on success in an introductory university course (Ahadi *et al.*, 2017). Other factors that may affect course success have been studied but not in depth (Baldwin *et al.*, 2017). Two recent studies have shown a positive relationship between mathematics or science background to computer programming success (Farmer & Tierney, 2017). Various factors regarding student learning styles and learning to the program have been found in many studies (Bain & Wilson, 2017). Other interesting factors that have been addressed in recent studies include student attributions of success to oneself or to outside forces (McGee *et al.*, 2017) and students' course outcome expectations (McGee *et al.*, 2017). A factor of potential interest that has not widely been

studied in computer programming is computer playfulness (Horton & Craig, 2015). A negative factor affecting student success is a high amount of game playing by students (Huang & Shiau, 2017).

Self-Efficacy and Its Role in Learning

Holzberger *et al.* (2013) define self-efficacy, as “...people’s judgments of their capabilities to organize and execute courses of action required attaining designated types of performance.” Self-efficacy beliefs are keys element in human performance over a very broad range of situations, for example, efficacy for work tasks, for physical activities, and for personal relationships (Schultz & Schultz, 2016). Self-efficacy is important in learning activities because learning involves more than just acquiring skills. As Holzberger says, “...competent functioning requires both skills and self-beliefs of efficacy to use them effectively” (Miller & Ramirez, 2017). Schultz and Schultz (2016) are of the opinion that learning situations and self-efficacy influence the use of cognitive strategies while solving problems. The amount of effort expended, the type of coping strategies adopted, the level of persistence in the face of failure, and the ultimate performance outcomes are all influential in determining computer programming proficiency. Chamorro-Premuzic (2016) states that according to self-efficacy theory, judgments of self-efficacy are based on four sources of information. These are the individual’s performance attainments, experiences of observing the performance of others, verbal persuasion, and physiological reactions. The most important is performance attainments, that is, the individual’s evaluation of the outcomes of his or her direct attempts to perform an activity.

Educational researchers recognize that, because skills and self-beliefs are so intertwined, one way of improving student performance is to improve student self-efficacy. Interventions to improve student self-efficacy focus on specific skills or knowledge and target the four sources of information that students use to evaluate their self-efficacy, as defined above. Providing students with direct hands-on experiences in an activity is critical since the strongest source of information is performance outcomes (De Neve & Ro, 2015). Making positive hands-on experiences is also important, especially in the early stage of learning, when the task may seem overwhelming. According to Schiefele (2017) attempts have also been made, with some success, to increase self-efficacy in learning by peer modeling of tasks, verbal persuasion, or other types of social influences, such as cooperative learning environments (Phillips *et al.*, 2017).

Mental Models and Computer Programming

Guzdial *et al.* (2017) define a Mental Model as a predictive representation of real-world systems. People create internal representations of objects and processes in the world, and they use these mental representations to reason about, explain and predict the behavior of external systems. Mental models are critical in debugging a process when things go wrong because the mental model supports the person in reasoning about and localizing possible faults (Moyer *et al.*, 2017). Mental models have been studied in many domains and situations.

In recent years, the mental model's concept has been popularized by practitioner magazines and websites in areas such as human-computer interaction (Huang & Shiau, 2017). Programming is a

cognitive activity that requires the programmer to develop abstract representations of a process and express them in the form of logic structures. In the case of creating, modifying, reusing, or debugging a program, the programmer must also translate these abstract representations into completely correct code using a formal language. Having a well-developed and accurate mental model is likely to affect the success of a novice programmer in an introductory programming course (Moyer *et al.*, 2017).

A programmer's mental model could encompass useful knowledge about how programs work in general, stereotypical ways of solving common programming problems and how a particular program is structured and functions, as well as knowledge about the syntax and semantics of a specific language (Coles and Phalp (2016).

Chen (2017) referred to mental models as schemas or plans that have been showing to play an important role in program comprehension and in comprehension-related tasks, such as modification and debugging. Rumsey *et al.* (2017) found strong effects of mental model formation in a program modification task. Participants were asked to modify a program but not given any explicit instructions about how to approach the task. The results showed that programmers who first attempted to systematically read and comprehend the program were much more successful in doing the modifications than programmers who jumped immediately into making modifications.

The difference in performance between programmers who built a mental model of the program and those who did not be especially great in modifications that involved interactions with code in other parts of the program. Similar results were reported by Schoeman and

Gelderblom (2016) in a comparison of novices and experts debugging a program. A conclusion can be made from these studies is that novices' success in programming tasks may be increased by greater attention to building a good mental model of the program. These studies of mental models in programming do not deal directly with the issue of success in introductory programming courses. However, the various factors a good mental model and success in programming tasks suggest that having a good mental model may be an important contributor to course outcomes.

2.4 The Study's Theoretical Model

This study adopts a model of computer programming performance by novice programmers based on the factors of Previous Experience, Self-efficacy, and Mental Model. The study's theoretical model has been adapted from the Model of Factors that Influence the Learning of Computer Programming (abbreviated as MFILCP) taken from Wiedenbeck *et al.* (2004). The adapted model is been illustrated in Figure 2.1. The main constructs or variables of the model are been represented by ovals and the relationships between these constructs are illustrated as directional lines.

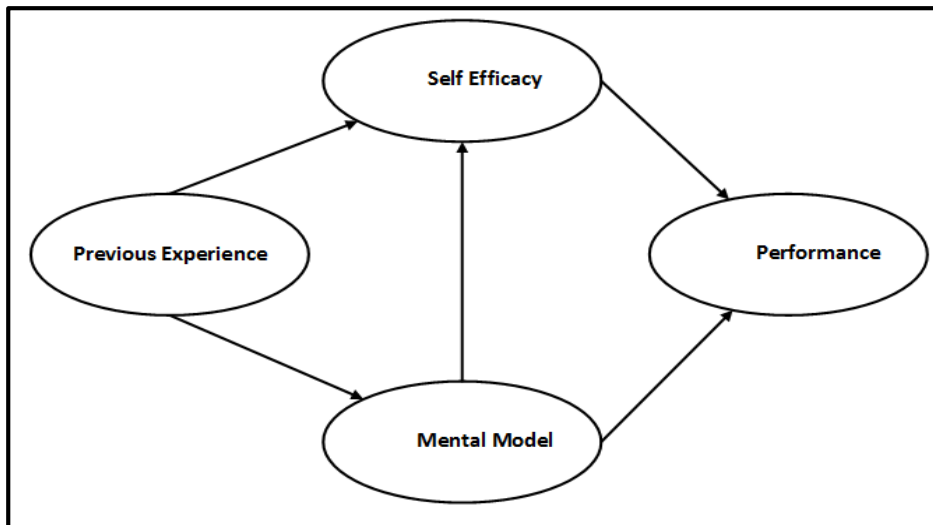


Figure 2.1: The MFILCP adapted from Wiedenbeck *et al.* (2004)

Previous experience: Previous experience is important to success in an introductory programming course. Previous experience acts as a significant predictor of both students’ self-efficacy and mental models of programming, which in turn predicts course performance (Baldwin *et al.*, 2017)

Self-efficacy: Based on self-efficacy theory, there is a positive, causal relationship between previous experience and self-efficacy in the domain of application. An extrapolation of this relationship into the domain of computer programming suggests that as the ‘hands on’ exposure to computer programming tasks increase, so too will students’ self-efficacy in computer programming increase. It is also hypothesized that students’ mental models of programming will have a significant effect on their self-efficacy beliefs (Miller & Ramirez, 2017).

Mental Model: Previous experience also has a positive influence on the accuracy of the mental models that students develop when trying to solve a computer programming task (Wiedenbeck *et al.*, 2004). A further observation is that a clear mental model of what programs do and how they do, it will increase students' feelings of self-efficacy about programming. It is expected that both the Mental Model

Performance: According to Wiedenbeck *et al.* (2004), performance in computer programming is influenced by previous experience. However, this relationship is not a direct one. A more accurate representation is that performance in computer programming is linked to previous experience through the mediating influence of Self-efficacy and Mental Model. According to Mason (2017), self-efficacy explains is a pivotal factor when it comes to understanding students' performance in computer programming courses (Mason, 2017).

2.5 Summary

The current chapter has provided detail on issues related to student performance in computer programming assessment. A significant outcome of this discussion is that students have been grappling with the mastery of computer programming at an introductory level. An incursion into the realm of cognitive processing that influences the acquisition of computer programming skill suggests that previous experience, self-efficacy and mental models of the problem domain play a significant role in determining computer programming proficiency. A theoretical model aligned with the preceding constructs

has been identifying as a viable framework to underpin the empirical phase of the current study.

The subsequent chapter describes the methodology, the research approach, choice of data collection methods, the study site, the study's sample, and the data collection instrument.

3.0 THE RESEARCH METHODOLOGY

3.1 Introduction

The literary incursion into factors that influence the learning of computer programming has been discussed as part of the literature study, the Model of Factors that Influence the Learning of Computer Programming (MFILCP) has been identified as the underpinning theoretical framework for the study. The MFILCP plays a defining role in the methodology adopted for the study.

Research is conducted for two reasons; either to find a solution to a problem or to answer a question (Marinova & Hartz-Karp, 2017). According to Creswell and Poth (2017), research is defined as a process of finding new information on a specific topic. However, research methodology refers to the process, tools, and procedures that are adopted when conducting a research study (Alvesson & Sköldbberg, 2017). It is further described as a systematic way and process that is adopted when carrying out a research study (Robson & McCartan, 2016).

The research process and procedures used in the current study has been informed by the Research Onion model presented in Saunders *et al.* (2017) and illustrated in Figure 3.1.

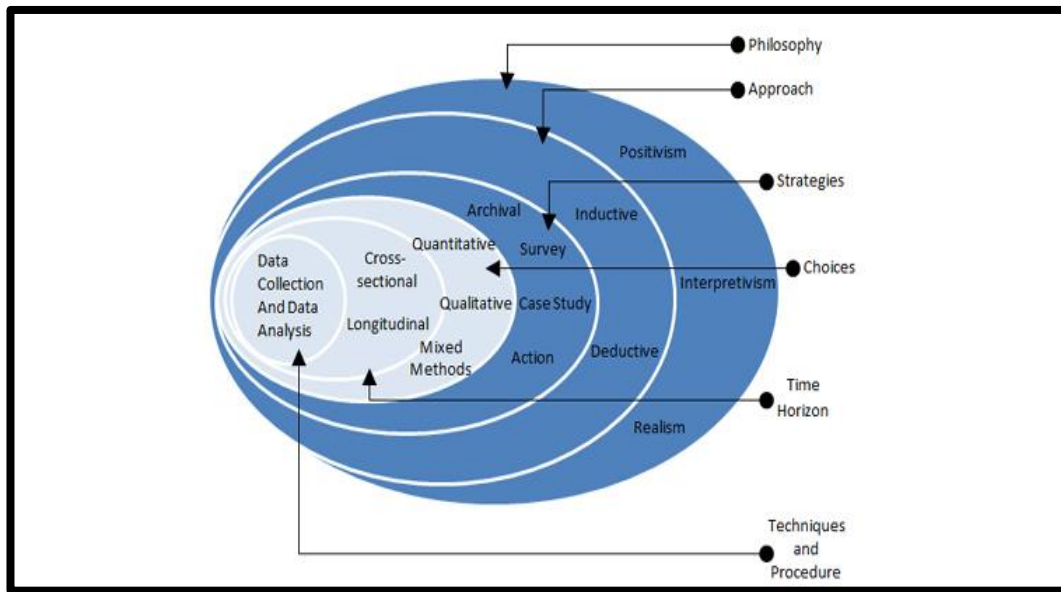


Figure 3.1 The research Onion taken from Saunders *et al.*, (2017, p. 54)

The process and procedures like the research philosophy, research approach; choice of data collection, time horizon, techniques and procedures adopted in this study is explained in the various sections of the current chapter to answer the following research questions:

- How does ‘previous experience’ influence students’ performance in the learning of computer programming?
- How does Self-efficacy influence students’ performance in the learning of computer programming?
- How does the ‘Mental Model’ representation of the problem domain influence students’ performance in the learning of computer programming?

- How does the ‘mental model’ representation of the problem domain influence students’ self-efficacy in the learning of computer programming?

3.2 Research Philosophy

Research philosophy refers to the researcher’s belief and philosophical approach adopted to investigate a phenomenon (Saunders & Tosey, 2013). The different types of research philosophies are positivism, realism, interpretivism, and pragmatism (Saunders & Tosey, 2013).

Realism is defined as a philosophical approach associated with scientific inquiry (Saunders & Tosey, 2013).

Interpretivism approach involves gathering rich and detailed insights into a phenomenon. It involves using small samples and it is best suited for a qualitative study (Saunders & Tosey, 2013).

Pragmatic approach, on the other hand, uses multiple techniques for gathering data and it results in an interpretation which is best suited for studies that adopt both a quantitative and qualitative method for better understanding of a problem (Saunders & Tosey, 2013).

A positivist approach is an approach that involves using scientific methods to test theories (Saunders & Tosey, 2013). It gathers data from a larger sample (Patten & Newhart, 2017). It also uses an empirical approach to addressing a research problem thereby providing an explanation based on what is observed thus giving room for generalization (Creswell & Poth, 2017).

The current study adopts a positivist research approach because it is quantitative in nature. Furthermore, this study follows an empirical approach by gathering data from a large sample and employs

statistical analysis in order to provide a comprehensive explanation of the previous experience of self-efficacy and the mental model to determine the performance of students at DUT

3.3 Research Approach

The research approach is largely dependent on the research philosophy that a researcher adopts to guide his / her study (Saunders & Tosey, 2013). The approach a study adopts depends on what a study intends to address. Research approaches are of two types, a deductive and an inductive research approach (Creswell & Poth, 2017). An inductive approach is the opposite of a deductive approach. In an inductive research approach, the researcher develops the research questions before adopting a model and it is most suitable for qualitative research (Saunders & Tosey, 2013)

In a deductive research approach, the researcher firstly examines the previous literature conducted on the study, then extracts variables considered in the literature and the model that guides the study (Creswell & Poth, 2017). The research questions and the variables considered in a deductive approach are developed from the literature examined and the model adopted to guide the study (Saunders & Tosey, 2013). A deductive research approach uses the scientific method for data collection which is analyzed using statistical analysis (Anderson-Gough *et al.*, 2017). It aligns with a positivist research philosophy because it is highly objective in nature, and it is best suited for studies, which attempt to explain various factors (Saunders et al., 2017).

The current study embraces a deductive approach because the model adopted guided the research questions as well as the variables

considered in this study in order to investigate the factors influencing learning computer programming

3.4 Research Strategy

Research strategy refers to procedures adopted by researchers when addressing research questions (Saunders & Tosey, 2013). The different strategies involved in conducting a research study are experimental, survey, case study, action, grounded, ethnography and archival research strategy.

An experimental research survey is mostly adopted when comparing the effect of a phenomenon on two different groups; a controlled and a treatment group (Saunders & Tosey, 2013). A case-study research strategy involves conducting a study by selecting a certain subject with a certain characteristic (Saunders & Tosey, 2013). Action research is usually adopted when attempting to find a solution to a problem identified (Creswell & Poth, 2017). In ethnography research, the researcher conducts the study in the context of a specific culture or group, while an archival research is a strategy where the researcher obtains data from existing data; usually involving secondary data (Creswell & Poth, 2017).

A survey strategy gives room for the researcher to collect data from respondents which is a representation of the whole population hence giving room for objectivity (Alvesson & Sköldbberg, 2017). In other words, “survey strategy presents the opinions of a population by studying a sample of that population” (Creswell & Poth, 2017). A survey strategy mostly involves the use of a questionnaire to gather data from

a sample. In other words, it follows the deductive approach (Jagdale *et al.*, 2018)

The current study adopts a survey research strategy because a sample was taken from a part of the whole population of the university students to investigate the factors influencing the learning of computer programming.

3.5 Research Design

According to Creswell and Poth (2017), a research design gives the direction of a research study. It also articulates what method(s) a research study will adopt (Glaser, 2017). There are three major types of research designs: descriptive, exploratory and explanatory research design.

An exploratory research design is best suited for studies that provide a deeper insight into a research problem that is not clearly understood (Glaser, 2017), while a descriptive research design simply provides a comprehensive discussion of a problem that is being investigated (Coolican, 2017).

An explanatory research design, on the other hand, describes various factors by providing a detailed explanation of various factors in a study (Coolican, 2017). An explanatory research design gives detailed information compared to a descriptive research design (Saunders *et al.*, 2017).

Descriptive studies report instant data such as measures of central disposition including the mean, median, mode, deviance from the mean, variation, percentage, and correlation between variables, as descriptive study might employ methods of analyzing correlations

between multiple variables by using tests such as Pearson's correlation, regression, or multiple regression analysis.

The current study embraces a descriptive research design approach in order to study the factors influencing learning introductory computer programming at DUT.

3.6 Research Choice

Research choice refers to the mode of data collection when conducting a research study (Glaser, 2017). The different mode of data collection is mono-method, mixed-methods and multi-method (Saunders and Tosey, 2013).

A mixed-method uses a “combination of both qualitative and quantitative mode of data collection technique” (Creswell and Poth, 2017: p 38). A multi-method uses a combination of more than one quantitative and more than one qualitative mode of data collection technique (Saunders and Tosey, 2013). A mono-method uses only a single mode either of data collection technique, which can be using a questionnaire or by an interview (Saunders and Tosey, 2013).

The current study adopts a mono-method mode of data collection technique because it only gathered data with a questionnaire.

3.7 Time Horizon

Time horizon refers to the period it takes for a researcher to gather data. According to Saunders and Tosey (2013), there are two types of time horizon, they are a cross-sectional and a longitudinal time horizon.

The longitudinal time horizon research requires a researcher to gather data over a long period and it is suitable for an experimental and archival research strategy (Kim et al., 2017). The cross-sectional time horizon research requires data to be collected over a short period, and data is collected only once for the study (Saunders and Tosey, 2013). Cross-sectional research studies are most suitable for studies that adopt a case study and survey research strategy.

The current study embraces the cross-sectional time-horizon approach because it adopts a survey research strategy and data collection was carry out over a short period.

3.8 Research Site and Setting

The research site for this study is the Durban University of Technology, while the research setting is the Faculty of Accounting and Informatics on the Ritson Campus of DUT targeting the first year Bachelor of Technology (BTech) First-year students.

3.9 Population of the Study

A total population of 200 students was selected from the IT department at the Ritson campus of DUT to participate in the survey. According to Leon-Garcia (2017), sampling as a process of selecting a subset from a population. It also described as the process of selecting subjects for a study. A sampling technique is a process of selecting a member from a population for a study (Abdelkader, 2017).

There are two types of sampling techniques, namely probability and non-probability sampling technique. In a probability sampling technique, members of the population have an equal chance of participating in the study (Coolican, 2017), while in non-probability

techniques, “the members in the population do not have an equal chance of being selected to participate in the study” (Coolican, 2017: p 43).

An example of a probability sampling technique is a simple random probability technique where elements in the target population have an equal chance of selection to participate in the study (Elwood, 2017). Hence, this study utilizes a simple random probability technique because all the elements in the target population had an equal chance of selection to participate in the study

3.10 Sampling and Sampling Technique

The total sample size of this survey was calculated according to the sample size formula proposed by Naing et al. (2006) (see equation 3.1) for finite populations, where n = sample size, Z =confidence level, P =Estimated proportion, d =precision or acceptable margin of error, and N =Population size. The value of n was estimated using the following parameters: $Z=1.96$, $P=0.05$, $d=0.035$ and $N= 204$ students which give a sample size of 200 students. The construction of the sample for 200 students surveyed by this study was done as follows: The ratio of the students in that department from business analysis and application development were calculated compared to the total number of registered students for application development in the department, and this ratio was multiplied by the sample size in order to get the number of students in the sample for the department. Equation 3.1 was used to determine the sample size for the study.

Equation 3.1.

$$n = \frac{NZ^2P(1 - P)}{d^2(N - 1) + Z^2P(1 - P)}$$

According to Scott (2017) and Moser and Kalton (2017), the population of a study is a group of people that have a common characteristic, while a target population refers to the population that includes all clinical and demographic characteristics where results will be generalized.

The target population considered in this study was Bachelor of Information and Communication Technology (First year) students in the Faculty of Accounting and Informatics on the Ritson Campus of DUT. The Ritson Campus has the highest population of students who learn introductory computer programming at DUT. Also, first-year students are the ones likely to face factors influencing the learning of computer programming because of little experience compared to mature students who have experienced coding, debugging and implementation right from their undergraduate study, hence, they can provide robust information on how factors influenced their academic performance in computer programming over the years.

3.11 Data Collection Instrument

The data collection instrument refers to the tool used in collecting data in a research study (Saunders and Tosey, 2013). In a qualitative study, data is collected through observation and the use of interviews, which can be structured, semi-structured or unstructured, and are

mostly analyzed using content analysis (Berggren et al., 2017). A mixed method mode of data collection uses both interviews and a questionnaire (Saunders and Tosey, 2013). A mixed method data collection is suitable when either the quantitative or the qualitative approach is inadequate to address suitably the problem that is being investigated (Creswell and Poth, 2017).

A quantitative model of data collection, on the other hand, uses a questionnaire to gather data, which analyzed are and interpreted using a statistical test (Creswell and Poth, 2017). It uses more samples compared to a qualitative research, and it gives room for generalization of results. A quantitative research mostly deals with numbers and statistics and it used is to examine various factors i.e. independent and dependent factors in a study (Saunders and Tosey, 2013).

In the current study, data was collected using a questionnaire because it is a quantitative study. In addition, the result of the study was analyzed and interpreted using statistical analysis in order to examine the influence that the variables identified in the study, have on the learning of computer programming.

3.11.1 Questionnaire Design

The questionnaire was designed using Microsoft Word. It was administered physically to the target population identified in the study.

The questionnaire was designed to conform to the Psychology of Programming model that underpinned the study. These were done in order to get a well-informed response, which provided a richer and better understanding to explain the influence of factors influencing the

learning of introductory computer programming. The questionnaire was divided into four sections (Refer to Appendix B).

Section A consists of the demographic background information and previous experience. Questions asked in this section include participants' name, student number, age, gender, and Matric or any equivalent Merits. The two basic questions asked in this section also include participant experience. For example, did you ever write a computer program in any language before you started APPDev @ DUT this year? Is this your first course in programming, if not, what other programming courses have you studied.

Section B - The self-efficacy construct as seen in the Psychology of Programming model in Figure 4.1 is examined here. This section was to identify how familiar or skillful participants are in computer programming. This addressed Research Question 2.

Section C – The Mental model construct as seen in the Psychology of Programming model in Figure 4.2 is presented here. This section was to identify critical thinking to determine a mental model and the ability to maintain consistency in resolving basic computer programming problems. This addressed research question part three.

3.11.2 Data Quality Control

Data quality control ensured by conducting a reliability and validity test on the data collection instrument used in this study. Reliability refers to the extent to which a test or an experiment will yield the same results when carried out repeatedly (Montgomery, 2017). In other words, if a study can reproduce similar results under different circumstances, then it is considered reliable (Leon-Garcia, 2017).

Validity refers to how truthful the results of a study are. It also determines whether the research accurately measures what it is intended to measure (Montgomery, 2017).

The researcher ensured reliability by conducting a consistency test using the Cronbach's alpha coefficient test on the parameter used to measure academic performance in this study. Validity was also ensured by conducting a pilot study through the process of distributing questionnaire among 10 participants identified in the target population. The feedback received was used to modify the questionnaire in order to remove aspects that were ambiguous or not clear. An adjusted questionnaire was designed which conveyed questions in a simple language so that it could be easily interpreted by participants. Also, the researcher ensured that all questions asked were strongly aligned with the objectives of the study.

3.11.3 Ethical Considerations

The researcher requested permission from the registrar of the University in order for the study conducted to be among students at the Durban University of Technology (DUT). After the permission was granted by the registrar, the researcher attached a copy of the permission letter to a copy of the questionnaire (Refer Appendix C) and a duly filled in ethical form was forwarded to the Ethics Committee of the University in order to be granted ethical approval for the study to be conducted.

An ethical approval letter issued to the researcher (Refer Appendix C), which granted the researcher permission to conduct the study among students of the university. A consent form attached was

the questionnaire where participants either accept or decline to participate in the study.

The researcher explained the purpose of the study to the target population, stating the objectives of the study, emphasizing that participation is voluntary and both the researcher and the school will uphold confidentiality. The participants assured that their identity would not be revealed without getting their approval should there be a need for it to be revealed. Also, data collected will be used only for the purpose of this study.

3.12 Summary

This chapter has described and justified the methodology adopted to explore the factors that influence first-year student's learning introductory computer programming language at DUT. The research philosophy, research approach, research survey, research design, and the choice of data collection used for this study were justified and explained. The sampling technique adopted, the data collection instruments used and the procedure for data collection were all described in details.

The following chapter presents the results and analysis of the data gathered in this study.

4.0 DATA PRESENTATION, AND DISCUSSION OF THE FINDINGS

4.1 Introduction

The previous chapter provided a discussion on the research methodology that has been implemented to undertake the empirical phase of the current study. The current chapter provides detail of the data analysis that has been conducted to facilitate the answering of the study's main questions and to ensure that the objectives of the study have been achieved. The presentation and analysis of the findings were based on the research, which objectives are:

- To understand the influence of previous experience on students' learning of introductory computer programming;
- To understand the influence of self-efficacy on students' learning of introductory computer programming;
- To understand the influence of the 'mental model' representation of the problem domain on students' learning of introductory computer programming.

The data are presented in three parts. The first part of the chapter presents the results using frequency and percentage graphs. The second part deals with the presentation of the results using descriptive statistics. The third part covers the presentation of the results using inferential statistics. As indicated in Chapter 3, 200 respondents were selected for the study. The researcher administered 200 questionnaires

to the respondents and all the 200 questionnaires were retrieved from the respondents, which represented a 100% response rate.

4.2 Reliability: Cronbach's Alpha Coefficient

Reliability measures the degree of the consistency of the research instrument over time. In other words, it is the extent to which the research instrument measures what it was designed to measure (Sekara & Bougie, 2013).

The Cronbach's alpha coefficient was computed to determine the reliability of the research instrument used. The Cronbach's alpha coefficient of 0.70 and beyond was considered as reliable as recommended by Sekaran and Bougie (2013). The results are shown in Table 4.1.

Dimensions	No.	Cronbach's Alpha
Self-efficacy	9	0.901

Table 4.1 Reliability: Cronbach's Alpha Coefficient

Table 4.1 indicates that the questionnaire for measuring self-efficacy has a very high degree of inter-item consistency and reliability ($\alpha = 0.901$). Therefore, the questionnaire measuring self-efficacy is reliable and can be used by other researchers for the same purpose.

4.3.1 KMO and Bartlett's Test

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		0.903
Bartlett's Test of Sphericity	Approx. Chi-Square	2922.745
	Df	36
	Sig.	0.000

Table 4.2 KMO and Bartlett's Test Measure Sampling Adequacy

It is evident from the Table 4.2 that Measure of Sampling Adequacy [MSA] for the self-efficacy, a mental model is 0.903, and Bartlett's test is significant, which indicates that the data set complies with the requirements of sampling adequacy and sphericity for the factor analysis performed.

4.4 Overview of the Questionnaire Design

The questionnaire (Appendix A) was designed to conform to the Model of Factors that Influence the Learning of Computer Programming (MFILCP) proposed by Wiedenbeck *et al.* (2004). The item in each section of the questionnaire was aligned with the constructs of the model adopted for this study. The objectives of these constructs are explained in Table 4.3.

The Questionnaire Design

Section	Constructs of the model	Objective of question	Question number in the questionnaire	Total
---------	-------------------------	-----------------------	--------------------------------------	-------

A	Previous Experience	To understand the previous experience factors that influence students' learning of computer programming	A4, A5	3
B	Self-efficacy	To understand the influence of self-efficacy factors on students' learning of computer programming	B1- B9	9
C	Mental Model	To understand the influence of mental model factors on students' learning of computer programming	C1- C4	4

Table 4.3: Overview of Questionnaire Design

Responses were recorded on a 5-point Likert scale ranging from 'very easy' to 'very difficult'. For analysis purposes, 'very easy' was coded as five and 'very difficult' was coded as one. A neutral response was coded as three. The analysis of the findings were represented in figure and tabular format indicating frequency and percentage as participants' responses to each of the four categories from the measure of Previous Experience, Self-efficacy and to that of Mental Model responses were

recorded on a 3-point assignment scale ranging from ‘correct answer’, “almost correct“ and ‘wrong answer. For analysis purposes, ‘correct answer’ was coded as three, ‘almost correct’ was coded as one and ‘wrong answer’ was coded as zero.

4.5 Descriptive Statistics

The perceptions of the participants regarding self-efficacy and mental model were assessed by asking the respondents to respond to various aspects of the items using a 1 to 5 point Likert scale and 1 to 4 point Likert scale for SE respectively and to that of Mental Model responses were recorded on a 3-point assignment scale ranging from 0 to 3 point marks assignment. The results were processed using descriptive statistics. Descriptive statistics is a statistical tool which is used to summarize or describe numerical data (Wilson, 2010). The purpose of using descriptive statistics is to inform the readers on the overview of the data gathered prior to the data analysis. In this study, the kind of descriptive statistics employed is mean, standard deviation, minimum and maximum.

The mean is also known as the arithmetic average of a frequency distribution (Wilson, 2010). The mean was determined through the summation of the individual items in the questionnaire and then divided by the total number of the items. It was computed using the SPSS. The mean helps to identify the strength and the direction of the value. Using the scale of 1-5, the mean score of 3 and above is considered as significant while below 3 is non-significant. Also, using the scale of 1-

4, the mean score of 2 and above is considered as significant, while below 2 is considered non-significant.

According to Saunders et al. (2009), the standard deviation is used to describe or compare the extent to which the data value for a variable is spread around the mean. Sekaran and Bougie (2013) suggested that the standard deviation is a commonly used measure of dispersion, being a square root of the variance which indicates the range of variability in the data. A large or positive standard deviation indicates that the data values are far from the mean while a small or negative standard deviation indicates that values are clustered closely around the mean. The sign of the deviation (positive or negative), reports the direction of that difference. Both the maximum and minimum are normalization methods. The maximum represents the highest scale while the minimum represents the least scale. For example, on the scale of 1-5, the maximum is 5 while the minimum is 1. The results of the descriptive statistics are shown in Table 4.4.

4.5.1 Demographic information

This section describes the demographic profile of the respondents, including age, qualification obtained, programming experiences and performance. The findings are presented in Table 4.4.

Information	N	%
Gender		
Male	143	71.5
Female	57	28.5
Other	--	0.0

Age			
	17-20 Years	127	63.5
	21-23 Years	54	27.0
	24 Years and Above	19	9.5
Experience in computer programming			
	Yes (2-5 years)	135	67.5
	No (less than 2 years)	65	32.5
Programming year experience			
	1-3 Months	70	35.0
	4-6 Months	32	16.0
	7-11 Months	--	0.0
	1 Year	15	7.5
	2 Years	6	3.0
	Other	77	38.5

Table 4.4 Previous Experience

As shown in the Table 4.6 7.15% of the respondents representing the majority were males. The majority (63.5%) of the respondents were between the ages of 17- 20 years. Furthermore, 67.5% respondents had experience in computer programming. Besides, 38.8% of the respondents who constituted the majority had experience in the C# programming. Also, a significant observation was that 38.5% of the respondents had no experience in computer programming.

4.5.2 Self efficacy

There are nine (9) items, which measured self-efficacy in the study. Self-efficacy was measured using the Computer Programming Self-efficacy scale (Chen, 2017). This instrument used previously by

Kunkle and Allen (2016) in their research on success factors in introductory computer programming courses. The scale consists of nine questions that ask students to judge their capabilities in a wide range of programming tasks and situations. As mentioned earlier, the study utilizes the 5-point Likert scale ranging from ‘very easy’ to ‘very difficult’. The findings are shown in Figure 4.1.

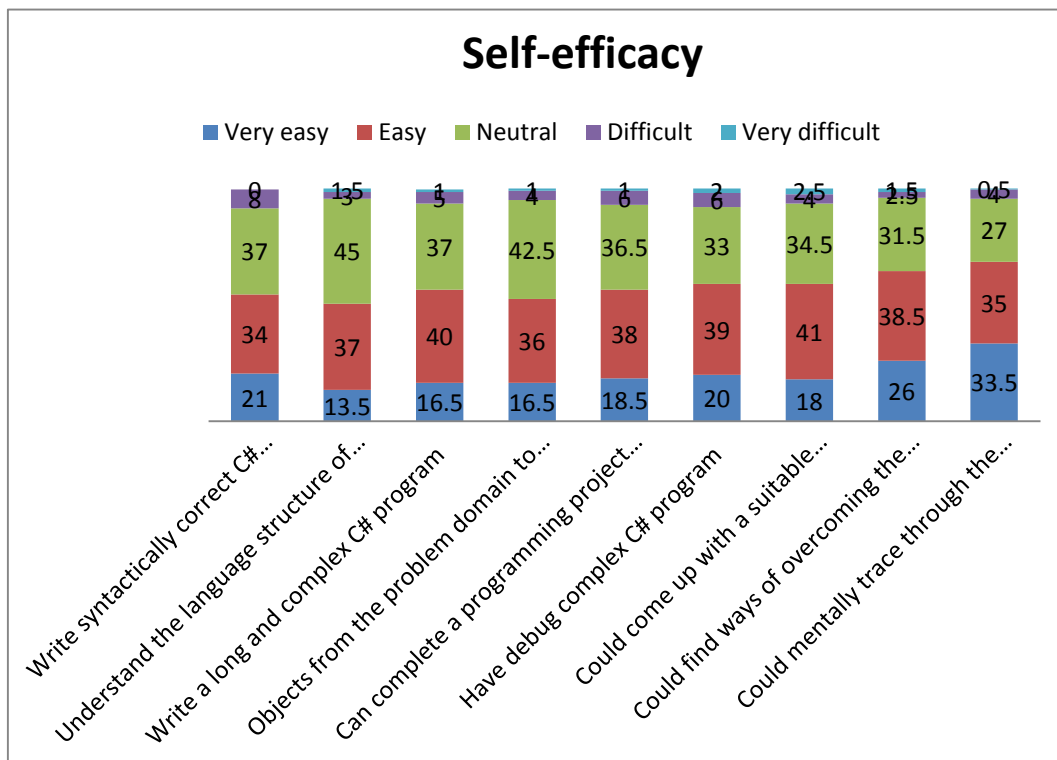


Figure 4.1 Self-efficacy

The information from Figure 4.1, shows that the majority (21% + 34% = 55%) of the respondents reported that it was easy for them to write syntactically correct C# statements. Approximately half (50.5%) of the respondents indicated that they easily understood the language structure of the C# programming language. Also, 56.5% reported that it

was easy for them to write a long and complex C# program to solve any given problem. From an object-oriented perspective, approximately 52.5% of the respondents were of the view that it was easy for them to relate objects from the problem domain to their C# application. The results showed that 56.5% of the respondents reported that they could complete a programming project once someone else helps them get started. Another 59% of the respondents indicated that they have debugged (correct all the errors) complex C# program that they had written and made it work. Furthermore, 59% of the respondents were of the view that they could come up with a suitable strategy for a given programming project in a short time. The findings revealed that 64.5% of the respondents reported that they could find ways of overcoming the problem if they got stuck at a point while working on a C# programming project. Besides, 68.5% of the respondents indicated that they could mentally trace through the execution of a long C# program.

4.6 Mental Model

The researcher made use of the words “Correct answer” “partially correct” and “wrong answer” in this context for data analysis purpose to categorizes those respondents who provide the right answers as requested by the researcher and points are been assigned on each multiple choice answers in assending order from 1 to 3 i.e wrong answer =1, almost correct =2 and right answer =3. The scope of these points assignment is to give higher marks to those students who answered the question correctly and also to categorize those students who almost got

the right answer from those who got it wrong, with breaking down of the section data will be easily interpreted in understandable manner in Figure 4.2.

The mental model questions are designed to be a mental exercises with the option of the answers very familiar to each other, as researcher with programming knowledge knowing fully well that students always found problem solving task challenging. therefore, students must be very sure of their solution before selecting an answer of their choice, each question are subsequently designed to be familiar to each other and likewise the options of right or wrong answers but required thorough solution in other to identify that which is the correct answers from other similar option known as partial answer.

This section of the study is “driven” by the programming related tasks that the respondents of the study were asked to respond to the programming related tasks were focused on assignment statements that entailed an interchange of primitive and object-oriented variables (as can be viewed in Table 4.5). The researcher expected that students would have some notion of what $a = b$ might mean, and would use that knowledge in providing a correct response to the question asked. Each of the questions asked in Table 4.5 is linked to a different mental model as identified by the MFILCP theoretical framework that underpins the study. These mental models have been labeled as MM1, MM2, MM3, and MM4. The strategy used in the study was to allocate points/marks to the respondent’s answers depending on whether they provided a

correct answer for each of these mental model oriented questions. Table 4.5 provides a summary of the mental models.

Model	Description model
MM1	Person a, b, c; a = new Person (“Jack”); b = new Person (“Tom”); c = new Person (“Jim”); b = a; a = c; c = b; what is the value of b, a, c?
MM2	int a = 5; int b = 3 int c = 7; a = c; b = a; c = b; what is the value of a b c?
MM3	Person a, b; a = new Person (“Jack”); b = new Person (“Tom”); b = a; a = b; what is the value of b & a?
MM4	int a = 10; int b = 20; a = b what is the value of a=?

Table 4.5 The mental models identified in the study

As seen in Table 4.7, MM1 alludes to three objects identified as a, b and c. A means “Jack”, B means “Tom” and C mean “Jim”. From the mathematical point of view, $b = a; a = c; c = b$. Therefore, what is the value of a, b & c? In relations to MM2, $\text{int } a = 5; \text{int } b = 3 \text{ int } c = 7; a = c; b = a; c = b$; what is the value of a b c? Regarding MM3, Person a, b; a = new Person (“Jack”); b = new Person (“Tom”); b = a; a = b; what is the value of b & a? In reference to MM4, $\text{int } a = 10; \text{int } b = 20; a = b$ what is the value of a=? The results of the study are shown in Figure 4.2.

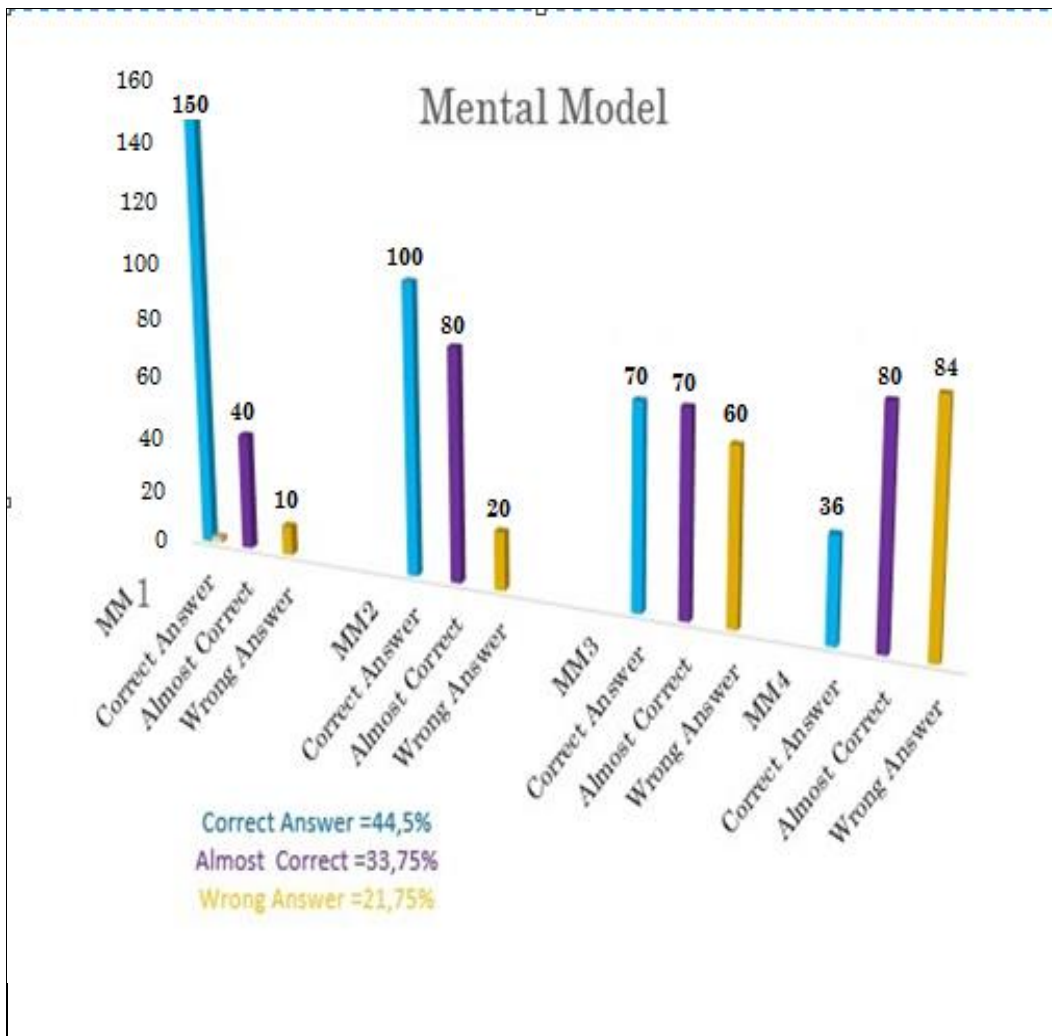


Figure 4.2 Number and percentage of participants for each model of value assignment

Figure 4.2 presents the rating of the student's mental model influence contributing to the factors affecting learning introductory computer programming at DUT. From the Figure 4.2, 44.5% of the participants representing the majority of the students have provided correct answers and 33.75% of the participants representing the almost answered correctly while 21.75% of them reported that they were unable to provide correct answers. Regarding the Mental Model Question 1, the

majority (150) of the participants answered correctly (e.g. Jack only), another 40 of the respondent's answers were deemed to be almost correct, while the remaining 10 responded with an incorrect answer.

In terms of the Mental Model Question 2, half (100) of the respondents answered the question correctly. Furthermore, 80 of the respondents provided an almost correct answer. However, the rest of the 20 of the respondents indicated that they answered the question wrongly.

In relation to the Mental Model Question 3, 75 of the respondents answered the question correctly. Another 75 provided an answer that was deemed to be almost correct. The remaining 60 of the respondents answered the question wrongly.

Concerning the Mental Model Question 4, 36 of the respondents answered the question correctly. Another 80 respondent provided an answer that was deemed to be almost correct. The remaining 84 of the respondents answered the question wrongly. From the findings, one could draw a conclusion that each of the respondents had an additional responsibility that could affect his/her study such.

Descriptive statistics: Key dimensions of the study

Dimension	Mean	95 % Confidence Interval		Std. Dev.	Min	Max
		Lower Bound	Upper Bound			
Self-efficacy	3.290	2.1981	2.3819	0.24653	1.00	5.00

Table 4.6 Descriptive statistics: Key dimensions of the study

Furthermore, on the scale of 1 to 4, a mental model had the mean score value of 2.455. The mean score suggests that there was a significant relationship between Mental Model and the performance of the students in computer programming.

4.7 Inferential Statistics

The inferential statistics were computed on the dimensions such as self-efficacy in order to assist the researcher to draw valid conclusions.

The Cronbach's alpha coefficient was further computed to determine the reliability of the research instrument used. The Cronbach's alpha coefficient of 0.70 and beyond what is considered as reliable. The results are shown in Table 4.6.

4.8 Spearman Correlation

Spearman correlation was computed to determine the relationship between the variables such as mental model, self-efficacy and performance of the respondents (students). The decision to opt for a Spearman Correlation is based on the observation that most of the study's data may be regarded as ordinal and the Spearman Correlation is a more robust test as compared to the Pearson Correlation test. The Spearman coefficient is measured on a scale with no units and can take a value from -1 through 0 to $+1$. If the sign of the correlation coefficient were positive, then a positive correlation would have existed. On the other hand, if the sign of the correlation coefficient is negative, then a negative correlation would have existed, indicating that those factors with a smaller number of a factor or its relationship were associated with

a larger variable, or vice versa. Table 4.7 below contains the results on Spearman correlation.

The strength of the correlation was based on the distance matrix from +1 or -1, meaning the closer the value to 1, the stronger the correlation (Archambault, 2002). Using Spearman's correlation analysis of the 3 factors (Previous Experience, Mental Model and Self-Efficacy), revealed the magnitude and direction of the association between the variables that are significant and positive as indicated in tables 4.7 to 4.9.

4.8.1 Spearman Correlation: Mental Model and Self-efficacy

Correlations				
			MM	SE
Spearman's rho	Mental model	Correlation Coefficient	1.000	.480**
		Sig. (2-tailed)	.	.000
		N	200	200
	Self-Efficacy	Correlation Coefficient	.480**	1.000
		Sig. (2-tailed)	.000	.
		N	200	200
**. Correlation is significant at the 0.01 level (2-tailed).				

Table 4.7 Mental Model and Self-efficacy

As reflected in the Table 4.7 above, there exists a significant positive relationship between mental model and self-efficacy. In other words, it can be explained that, there is a direct link between the mental model and self-efficacy suggesting that if students scored high on the self-

efficacy rating then in all likelihood, they will also have a correct mental model representation of the problem domain.

4.8.2 Spearman Correlation: Self-efficacy and previous - experience

Correlations				
			SE	PE
Spearman's rho	Self-Efficacy	Correlation Coefficient	1.000	.428**
		Sig. (2-tailed)	.	.000
		N	200	200
	Previous Experience	Correlation Coefficient	.428**	1.000
		Sig. (2-tailed)	.000	.
		N	200	200
**. Correlation is significant at the 0.01 level (2-tailed).				

Table 4.8 Self-efficacy and previous experience

As reflected in Table 4.10 above, there exists a significant positive relationship between mental models. In other words, it can be explained there is a direct link between self-efficacy and previous experience suggesting that if students scored high on previous experience then in all likelihood they will also achieve a high score on self-efficacy when it comes to computer programming.

4.8.3 Spearman Correlation: Previous Experience and Mental Model

Correlations				
			PE	MM
Spearman's rho	Previous Experience	Correlation Coefficient	1.000	.508**
		Sig. (2-tailed)	.	.000
		N	200	200
	Mental Model	Correlation Coefficient	.508**	1.000
		Sig. (2-tailed)	.000	.
		N	200	200
**. Correlation is significant at the 0.01 level (2-tailed).				

Table 4.9 Previous Experiences and Mental Model

As reflected in Table 4.9 above, there exists a significant positive relationship between previous experience and mental model. In other words, it can be explained there is a direct link between previous experience and mental model suggesting that if students scored high on previous experience, then they will also have a correct mental model representation of the problem domain.

4.9 Chapter Summary

The chapter presented, analyzed and discussed the key findings in line with the research objectives. The main research objectives were to

understand the influence of the main variables in the study, namely previous experience, self-efficacy and mental model representation of the problem domain on the learning of computer programming. Data was collected to operationalize these variables so that their influence on computer programming competency could be understood. By making use of Spearman's correlation, it has been established that each of the variables identified from the theoretical model used in the study, do have a positive influence on the learning of computer programming. The implications of these findings will be discussed in the subsequent chapter.

5.0 FINDINGS AND CONCLUSION

5.1 Introduction

- To evaluate the influence of previous experience on students' learning of introductory computer programming
- To evaluate the influence of self-efficacy on students' learning of introductory computer programming
- To evaluate the influence of the 'mental model 'representation of the problem domain on students' learning of introductory computer programming
- To evaluate the influence of the 'performance' representation of the problem domain on students 'self-efficacy in the learning of introductory computer programming

The results of the study are discussed and related to existing theory in sections 5.2 to 5.4 and the implications for teaching and learning are discussed in Section 5.5.

5.2 The influence of previous experience

The study investigated the influence of previous experience on students' learning of introductory to computer programming. The overall results of the study showed that the previous experience

positively influenced students' learning of introductory computer programming. Thus, there was a significant positive relationship between previous experience and student learning of introductory to computer programming. The findings from the study supported existing research on previous experience and students' learning of computer programming.

Scholarly literature shows that previous programming experience has a positive effect on success in an introductory university course (Ahadi et al., 2017). Another study showed that previous experience acts as a significant predictor of both students' self-efficacy and mental models of programming, which in turn predicts course performance (Baldwin et al., 2017). Recent studies have also shown a positive relationship between mathematics or science background to computer programming success (Farmer & Tierney, 2017). Various factors regarding student learning styles and learning to the program have been found in many studies (Bain & Wilson, 2017). Other interesting factors that have been addressed in recent studies include student attributions of success to oneself or to outside forces (McGee et al., 2017) and students' course outcome expectations (McGee et al., 2017). A factor of potential interest that has not widely been studied in computer programming is computer playfulness (Horton & Craig, 2015). A negative factor affecting student success is a high amount of game playing by students (Huang & Shiau, 2017).

5.3 The influence of self-efficacy on students' learning

The study further determined the influence of self-efficacy on students' learning of introductory computer programming at DUT. The response patterns showed that the respondents responded positively to all the items in the questionnaire. The analysis of the findings suggests that the respondents responded positively to all the nine items although there were some forms of disagreement among them. The findings suggest that the self-efficacy level among the respondents was high. The findings from this study are in keeping with existing research discussed in Chapter 2.

Self-efficacy has been considered as a very important aspect of learning activities because learning involves more than just acquiring skills, it also entails "...competent functioning that requires both skills and self-beliefs of efficacy to use them effectively" (Miller & Ramirez, 2017). Schultz and Schultz (2016) in their study argued that learning situations and self-efficacy influence the use of cognitive strategies while solving problems. They further suggested that the amount of effort expended, the type of coping strategies adopted, the level of persistence in the face of failure, and the ultimate performance outcomes are all influential in determining computer programming proficiency.

Similarly, Chamorro-Premuzic (2016) states that according to self-efficacy theory, judgments of self-efficacy are based on four sources of information, namely: individual's performance attainments, experiences of observing the performance of others, verbal persuasion and physiological reactions. However, the most important among the four is the individual's performance attainments. Educational

researchers recognize that, because skills and self-beliefs are so intertwined, one way of improving student performance is to improve student self-efficacy. Interventions to improve student self-efficacy focus on specific skills or knowledge and target the four sources of information that students use to evaluate their self-efficacy, as defined above. Providing students with direct hands-on experiences in an activity is critical since the strongest source of information is performance outcomes (De Neve & Ro, 2015). Making positive hands-on experiences is also important, especially in the early stage of learning, when the task may seem overwhelming.

5.4 The influence of the ‘mental model’ representation on students’ learning

The findings from the study suggested that mental model representation positively influenced students’ learning of introductory computer programming. According to Guzdial et al. (2017), a mental model is a predictive representation of real-world systems. The authors argue that people create internal representations of objects and processes in the world, and they use these mental representations to reason about, explain and predict the behavior of external systems. Moyer et al. (2017) postulate that mental models are critical in debugging a process when things go wrong because the mental model supports the person in reasoning about and localizing possible faults.

In recent years, the mental model's concept has been popularized by practitioner magazines and websites in areas such as human-computer interaction (Huang & Shiau, 2017). The scholars assert that programming is a cognitive activity that requires the programmer to

develop abstract representations of a process and express them in the form of logic structures. They suggest that when creating, modifying, reusing, or debugging a program, the programmer must also translate these abstract representations into completely correct code using a formal language.

5.5 Implications for Computer Programming Pedagogy

The current study provides empirical support for the validity of the main constructs from the theoretical model used in the study.

Previous Experience

Although the influence of previous experience has been commonly accepted as influential in the learning of computer programming, this commonly held perception has been confirmed in the context of students who learn computer programming at DUT. The pivotal role that previous experience plays needs to be recognised and mitigated in the case where students who enroll for computer programming courses and have little previous experience in the learning of computer programming. A viable strategy would be to add in “bridging courses” that enable students who have little prior experience of computer programming to use these course so that enhance their levels of programming experience. Clearly, the study has confirmed that such an intervention would result in better programming performance and correct mental model representations of the problem domain.

Self-Efficacy (SE)

The items used in the study’s questionnaire attest to students’ level of Se when it comes to computer programming. These items refer

to computer programming aspects such as mastery of a computer programming language with regards to its syntax as well as the use of the language to enable problem solving. Many of the computer programming languages taught at DUT, such as Java and C# are not easy to understand from a syntax perspective. The lack of understanding and mastery of the syntax will also influence students' ability to use these languages as problems solving tools. A lack of syntax understanding and a lack of ability to use a computer programming language as a problem solving tool will result in low SE scores thereby resulting in incorrect mental model representations and low computer programming performance scores. Hence as an intervention, experienced tutors should be assigned with the task of providing tutorial sessions that focus on the mastery of syntax and the ability to use this syntactical knowledge to solve simple computer programming problems. Extensive exposure to such an intervention will ensure that the students SE in computer programming will be improved.

Another strategy suggested in Coles and Phalp (2016) is the use of social persuasion techniques. Students working together, especially if they have different levels of self-efficacy, are in a position where social persuasion takes place. A tutor can facilitate social persuasion in the classroom or online by forming workgroups of students with different levels of capableness and giving them tasks that promote the interaction of group members (Coles & Phalp, 2016).

Mental Model

The correct mental model representation of the problem domain is pivotal to ensure that students are able to excel in computer

programming. However, this aspect of computer programming pedagogy is the most abstract and not easily achieved. The best that could be suggested in this regard is to engage in a strategy that mitigates the influence of previous experience in the hope that such a strategy would lead to better mental model representations of the problem domain. Also, the employment of experienced teachers of computer programming to teach computer programming would be highly beneficial to students. The challenge of obtaining a correct mental model representation of the problem domain may be mitigated if the course teacher has a clear and correct mental model representation of the problem domain so that this message is conveyed to the class. A serious shortcoming in this regard is that the lack of experienced teachers and lecturers result in a lack of correct knowledge that is disseminated to the class. In many instances, higher education institutions employ lecturers on the basis of their research credentials rather than on their technical knowledge of computer programming. This strategy is seriously detrimental to ensuring that students acquire a correct mental model representation of the problem domain.

Another strategy that could be used to enhance Mental Model is aligned to the suggestion in Coles and Phalp (2016) who recommend a strategy of getting students to steadily carry out tasks of increasing difficulty until they have a history of solid attainments. Frequent but small hands-on programming activities would be likely to build the history of success more than less frequent, large assignments. For students to monitor their capableness, timely and sufficient feedback is necessary so that students are able to incrementally build a correct mental model representation of the problem domain.

Coles and Phalp (2016) also suggest a technique of peer modeling. In programming courses, peer modeling could be “live” in a classroom with a peer working through a problem while other students watch or it could be done by students viewing a video of a peer successfully planning and executing a programming task. In peer, modeling it is important that the viewers see the model confronting difficult situations and overcoming them. The modeling scripted should not eliminate struggle because the point is for students to see how obstacles are overcome.

5.6 Conclusion

The chapter discussed the results of the study. The chapter has concluded with a set of suggestions that are oriented around the main constructs used in the study. These suggestions provide an empirical basis for strategies that may be used to improve computer programming performance by students in an educational setting. It is imperative that studies such as the current one are conducted on a regular basis and in different contexts so that a proper platform for the pedagogy of computer programming is provided for students in academic institutions. The study has achieved its objective of understanding the influence of previous experience, self-efficacy and mental model representation of the problem domain on the learning of computer programming.

REFERENCES

- Abdelkader, A., Alexopoulos, A., & Kotagal, P. (2017). Ontogeny of Seizure Semiology: a longitudinal study (P1. 246). *Neurology*, 88(16 Supplement), P1-246.
- Ahadi, A., Lister, R., Lal, S., Leinonen, J., & Hellas, A. (2017). *Performance and Consistency in Learning to Program*. Paper presented at the Proceedings of the Nineteenth Australasian Computing Education Conference.
- Alvesson, M., & Sköldbberg, K. (2017). *Reflexive methodology: New vistas for qualitative research*: Sage.
- Anderson-Gough, F., Edgley, C., & Sharma, N. (2017). Qualitative data management and analysis software. *The Routledge Companion to Qualitative Accounting Research Methods*, 405.
- Bain, G. M., & Wilson, G. (2017). CONVERGENT PATHWAYS IN TERTIARY EDUCATION What makes our students succeed? *ACM Inroads*, 8(2), 37-40.
- Baldwin, D., Barr, V., Briggs, A., Havill, J., Maxwell, B., & Walker, H. M. (2017). *CS 1: Beyond Programming*. Paper presented at the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.
- Balfanz, R., Bridgeland, J. M., Bruce, M., & Fox, J. H. (2012). Building a Grad Nation: Progress and Challenge in Ending the High School Dropout Epidemic. Annual Update, 2012. *Civic Enterprises*.
- Barnes, D. J., Kölling, M., & Gosling, J. (2017). *Objects first with Java: A practical introduction using Bluej*: Pearson.
- Bawa, A. (2012). South African higher education: at the center of a cauldron of national imaginations. *social research*, 76(3), 669-694.

- Berggren, M., Ruiz-González, C., Niño-García, J. P., & Del Giorgio, P. A. (2017). Contrasting dynamics and environmental controls of dispersed bacteria along a hydrologic gradient.
- Bowlick, F. J., & Goldberg, D. W. (2017). Computer Science and programming courses in Geography Departments in the United States. *The Professional Geographer*, *69*(1), 138-150.
- Breese, S., Milanova, A., & Cutler, B. (2017). *Using Static Analysis for Automated Assignment Grading in Introductory Programming Classes*. Paper presented at the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.
- Bringula, R. P., & Aviles, A. D. (2017). Factors Affecting Failing the Programming Skill Examination of Computing Students.
- Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., & Riddell, A. (2016). Stan: A probabilistic programming language. *Journal of Statistical Software*, *20*, 1-37.
- Chamorro-Premuzic, T. (2016). *Personality and individual differences*: John Wiley & Sons.
- Chen, C.-L., & Cheng, S.-Y. (2012). *A study of misconceptions and missing conceptions of novice Java programmers*. Paper presented at the Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS).
- Chen, I.-S. (2017). Computer self-efficacy, learning performance, and the mediating role of learning engagement. *Computers in Human Behavior*, *72*, 362-370.
- Chi, T.-Y., & Berger, D. E. (2017). Computer Skill Acquisition: The Effects of Computer-aided Self-explanation on Knowledge Retention and Transfer.

- Coles, M., & Phalp, K. (2016). *Brain-type as a programming aptitude predictor*. Paper presented at the 27th Annual Workshop of the Psychology of Programming Interest Group-PPIG.
- Coolican, H. (2017). *Research methods and statistics in psychology*: Psychology Press.
- Creswell, J. W., & Poth, C. N. (2017). *Qualitative inquiry and research design: Choosing among five approaches*: Sage publications.
- Cutts, Q., Fincher, S., Haden, P., Robins, A., Sutton, K., Baker, B., Box, I., de Raadt, M., Hamer, J., & Hamilton, M. (2006). *The ability to articulate strategy as a predictor of programming skill*. Paper presented at the Proceedings of the 8th Australasian Conference on Computing Education-Volume 52.
- De Neve, W., & Ro, Y. M. (2015). *Efficient and effective human action recognition in video through motion boundary description with a compact set of trajectories*. Paper presented at the Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on.
- Edgcomb, A., Vahid, F., Lysecky, R., & Lysecky, S. (2017). *Getting Students to Earnestly Do Reading, Studying, and Homework in an Introductory Programming Class*. Paper presented at the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.
- Elarde, J. (2016). Toward improving introductory programming student course success rates: experiences with a modified cohort model to student success sessions. *Journal of Computing Sciences in Colleges*, 32(2), 113-119.
- Elwood, M. (2017). *Critical appraisal of epidemiological studies and clinical trials*. Oxford University Press.
- Farmer, S. M., & Tierney, P. (2017). Considering creative self-efficacy: Its current state and ideas for future inquiry. *The Creative Self: Effect of Beliefs, Self-Efficacy, Mindset, and Identity*, 23.

- Glaser, B. (2017). *Discovery of grounded theory: Strategies for qualitative research*. Routledge.
- Grover, S., Pea, R., & Cooper, S. (2016). *Factors influencing computer science learning in middle school*. Paper presented at the Proceedings of the 47th ACM technical symposium on computing science education.
- Guzdial, M., Li, B., & Riedl, M. O. (2017). *Game engine learning from video*. Paper presented at the international Joint Conference on Artificial Intelligence (IJCAI 2017).
- Hair Jr, J.F., Sarstedt, M., Ringle, C.M. and Gudergan, S.P., 2010. *Advanced issues in partial least squares structural equation modeling*. SAGE Publications.
- Hašková, A., Munk, M., & Zahorec, J. (2014). Assessment of Selected Aspects of Teaching Programming in SK and CZ. *Informatics in Education - An International Journal*, 13(1), 157-178.
- Hjorth, M. (2017). Strengths and weaknesses of a visual programming language in a learning context with children.
- Holzberger, D., Philipp, A., & Kunter, M. (2013). How teachers' self-efficacy is related to instructional quality: A longitudinal analysis. *Journal of Educational Psychology*, 105(3), 774.
- Horton, D., & Craig, M. (2015). *Drop, fail, pass, continue Persistence in cs1 and beyond in traditional and inverted delivery*. Paper presented at the proceedings of the 46th ACM Technical Symposium on Computer Science Education.
- Hosseini. (2017). Assessing Programming Behaviors Through Evidence-Centered Design.
- Hovland, C. I., & Lumsdaine, A. A. (2017). *Experiments on mass communication* (Vol. 4976). Princeton University Press.
- Huang, L. C., & Shiau, W. L. (2017). Factors affecting creativity in information system development Insights from a decomposition

and PLS-MGA. *INDUSTRIAL MANAGEMENT & DATA SYSTEMS*, 117(3), 496-520. 10.1108/IMDS-08-2015-0335

Idemudia, E. C., Dasuki, S. I., & Ogedebe, P. (2016). Factors that influence students' programming skills: a case study from a Nigerian university. *International Journal of Quantitative Research in Education*, 3(4), 277-291.

Jagdale, S. C., Hude, R. U., & Chabukswar, A. R. (2018). Research Methodology *Encyclopedia of Information Science and Technology, Fourth Edition* (pp. 6767-6778): IGI Global.

Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness*: Harvard University Press.

Kim, M., Bae, J. M., Choi, K. H., Jung, H. M., Kim, S. Y., Kim, G. M., ... & Lee, Y. B. (2017). Subsequent vitiligo after hematopoietic stem cell transplantation: A nationwide population-based cohort study from Korea. *Journal of the American Academy of Dermatology*, 76(3), 459-463.

Krejcie, R. V., & Morgan, D. W. (2017). Determining sample size for research activities. *Educational and psychological measurement*, 30(3), 607-610.

Kumar, V. A., & Laakso, M.-J. (2016). Cultural Issues That Affect Computer Programming: A Study of Vietnamese in Higher Education. *Asian Journal of Education and e-Learning (ISSN: 2321-2454)*, 4(02).

Kunkle, W. M., & Allen, R. B. (2016). The impact of different teaching approaches and languages on student learning of introductory programming concepts. *ACM Transactions on Computing Education (TOCE)*, 16(1), 3.

Law, K. M. Y., Lee, V. C. S., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, 55(1), 218-228. doi: 10.1016/j.compedu.2010.01.007

- Lakshmi, S., & Mohideen, M. A. (2013). ISSUES IN RELIABILITY AND VALIDITY OF RESEARCH. *International journal of management research and reviews*, 3(4), 2752.
- Lopez, M., & Whalley, J. (2008). *Relationships between reading, tracing and writing skills in introductory programming*. Paper presented at the Proceedings of the fourth international workshop on computing education research.
- Lumsdaine, A., Duckworth, R., Rapp, J., Bjorholm, T., Demko, J., McGinnis, D., ... & Goulding, R. (2017). Progress in magnet design activities for the material plasma exposure experiment. *Fusion Engineering and Design*, 124, 211-214.
- Marinova, D., & Hartz-Karp, J. (2017). Methods for sustainability: Introducing pathways to hope *Methods for Sustainability Research* (pp. 1-13): Edward Elgar Publishing.
- Mason, R. (2017). *Introductory Programming Courses in Australasia in 2016*. Paper presented at the Proceedings of the Nineteenth Australasian Computing Education Conference.
- Mathews, D. K. (2017). *Predictors of Success in Learning Computer Programming*. The University of Rhode Island.
- Mauer, R., Neergaard, H., & Linstad, A. K. (2017). Self-efficacy: Conditioning the entrepreneurial mindset *Revisiting the Entrepreneurial Mind* (pp. 293-317): Springer.
- McGee, S., McGee-Tekula, R., Duck, J., Greenberg, R. I., Dettori, L., Reed, D. F., Wilkerson, B., Yanek, D., Rasmussen, A. M., & Chapman, G. (2017). Does a taste of Computing Increase Computer science enrollment? *Computing in Science & Engineering*, 19(3), 8-18.
- Miller, A. D., & Ramirez, E. M. (2017). The influence of teachers' self-efficacy on perceptions: Perceived teacher competence and respect and student effort and achievement. *Teaching and Teacher Education*, 64, 260-269.

- Montgomery, D. C. (2017). *Design and analysis of experiments*. John Wiley & sons.
- Moser, C. A., & Kalton, G. (2017). *Survey methods in social investigation*. Routledge.
- Moghaddam, M. H. Y., Leon-Garcia, A., & Moghaddassian, M. (2017). On the performance of distributed and cloud-based demand response in the smart grid. *IEEE Transactions on Smart Grid*.
- Moyer, K., Bapat, A., Morrison, L., & Ellman, M. (2017). Are Internal Medicine Residents Meeting the Bar? Comparing Resident Knowledge and Self-Efficacy to Published Palliative Care Competencies (S777). *Journal of Pain and Symptom Management*, 53(2), 452-453.
- Naing, L. G., Winn, T. N., & Rusli, B. (2006). Practical issues in calculating the sample size for prevalence studies. *Archives of Orofacial Sciences*, 1 (1): 9-14.
- Patten, M. L., & Newhart, M. (2017). *Understanding research methods: An overview of the essentials*: Taylor & Francis.
- Phillips, A. E., Walker, C. G., Ehrgott, M., & Ryan, D. M. (2017). Integer programming for minimal perturbation problems in university course timetabling. *Annals of Operations Research*, 252(2), 283-304.
- Rex, K., & Roth, R. M. (1998). The relationship between computer experience and computer self-efficacy to performance in introductory computer literacy courses. *Journal of Research on Computing in Education*, 31(1), 14-24.
- Robson, C., & McCartan, K. (2016). *Real world research*: John Wiley & Sons.
- Rodrigo, M. M. T., Tabanao, E., Lahoz, M. B. E., & Jadud, M. C. (2009). Analyzing online protocols to characterize novice Java programmers. *Philippine Journal of Science*, 138(2), 177-190.

- Rumsey, C. A., Burke, Q., & Thurman, C. (2017). *Cracking the Code: Bringing Introductory Computer Science to a Charleston Middle School*. Paper presented at the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.
- Saunders, M., & Tosey, P. (2013). The layers of research design.
- Saunders, P. U., Rice, A. J., Thompson, K. G., Woods, A. L., Sharma, A. P., & Garvican-Lewis, L. A. (2017). Four weeks of classical altitude training increases resting metabolic rate in highly trained middle-distance runners. *International journal of sport nutrition and exercise metabolism*, 27(1), 83-90.
- Schiefele, U. (2017). Classroom management and mastery-oriented instruction as mediators of the effects of teacher motivation on student motivation. *Teaching and Teacher Education*, 64, 115-126.
- Schoeman, M., & Gelderblom, H. (2016). *The Effect of Students' Educational Background and Use of a Program Visualization Tool in Introductory Programming*. Paper presented at the Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists.
- Schultz, D. P., & Schultz, S. E. (2016). *Theories of personality*: Cengage Learning.
- Thomsett-Scott, B. (2016). The Librarian's Introduction to Programming Languages.
- Veerassamy, A. K., D'Souza, D., Lindén, R., & Laakso, M.-J. (2017). The Impact of Prior Programming Knowledge on Lecture Attendance and Final Exam. *Journal of Educational Computing Research*, 0735633117707695.
- Watson, K. (2013). *Beginning Visual C# 2012 programming*. Indianapolis, Ind: John Wiley & Sons.
- Wiedenbeck, S., Labelle, D., & Kain, V. N. (2004). *Factors affecting course outcomes in introductory programming*. Paper presented

at the 16th Annual Workshop of the Psychology of Programming Interest Group.

APPENDIX A: QUESTIONNAIRE

RESEARCH TOPIC: FACTORS INFLUENCING FIRST YEAR STUDENTS' LEARNING INTRODUCTORY COMPUTER PROGRAMING LANGUAGE AT DUT

M.Com (ISDN) Coursework

The discipline of Information Systems & Technology
School of Management, Information Technology &
Governance University of
KwaZulu-Natal (Westville Campus)

Researcher: Kelvin Osaji-Onalo (0613497946)

Supervisor: Dr. Sanjay Ranjeeth (033 260 5641)

INTRODUCTION

My name is Kelvin Osaji-Onalo, an M.com student in the Discipline of Information Systems and Technology, School of Management, IT and Governance at the University of KwaZulu-Natal, Westville campus, Durban, South Africa.

The purpose of this questionnaire is to gather information from you in order to understand the factors influencing first-year students learning introductory computer programing at DUT. The questions asked will enable me to gain insights on the prior exposure to ICT environment and how much the factors influenced the students' performance in programming skill.

Through your participation, and with the result of the survey, I hope to provide recommendations on how to improve students' academic performance through previous experience, self-efficacy, and mental model.

The following keywords would be frequently used: previous experience, self-efficacy, mental model and performance:

Previous experience: A relationship between student learning styles and programming styles.

Self-efficacy: refer to as “people’s judgments of their capabilities to organize and execute courses of action required attaining designated types of performance.

Mental model: could encompass useful knowledge about how programs work in general, stereotypical ways of solving common programming problems and how a particular program is structured and functions, as well as knowledge about the syntax and semantics of a specific language.

Performance: This simply refers to how a student has performed in his/ her studies at an educational institution.

The filling of the questionnaire should take about 10 – 15 minutes.



**COLLEGE OF LAW AND MANAGEMENT
SCHOOL OF MANAGEMENT, INFORMATION TECHNOLOGY
AND GOVERNANCE**

M.COM (ISTN) Coursework

Researcher: Kelvin Osaji-Onalo (0613497946)

Supervisor: Dr. Sanjay Ranjeeth (033 260 5641)

Research Office: Ms. M Snyman (031 260 8350)

Reference No: HSS/ 2126 / 017M

Respondent No: -

CONSENT

I, (full names of participant) hereby confirm that I understand the content of this document and the nature of the research project.

I also understand that I am at liberty to withdraw from the project at any time, should I so desire.

I hereby consent / do not consent to participate in this study.

SIGNATURE OF PARTICIPANT

DATE

.....
.....

FACTORS INFLUENCING FIRST YEAR STUDENTS' LEARNING INTRODUCTORY COMPUTER PROGRAMING LANGUAGE AT DUT

In those sections where options are provided, please indicate your response by making a cross (X) in the boxes provided.

SECTION A: Demographic Background Information and Previous experience

(1).Gender	Male <input type="checkbox"/>	Female <input type="checkbox"/>	Other <input type="checkbox"/>	
(2).Age	17-20yr <input type="checkbox"/>	21-23yrs <input type="checkbox"/>	24yrs and above <input type="checkbox"/>	
(3).Matric or any equivalent Merits obtained:	Matric <input type="checkbox"/>	(Specify)		
(4).Did you ever write a computer program in any language before you started APPDev @ DUT this year?	Yes <input type="checkbox"/> if yes what language?		No <input type="checkbox"/>	
	(1).C++ <input type="checkbox"/>	(2).C# <input type="checkbox"/>	(3).JAVA <input type="checkbox"/>	(4).(Specify) y)
(5). Is this being your first course in programming? If not, what other programming courses have you studied?	Yes <input type="checkbox"/>	Institution	Duration	Year
		(1) other varsity <input type="checkbox"/>	(1).1-3 weeks <input type="checkbox"/>	(Specify)
		(2) private colla <input type="checkbox"/>	(2). 2-6 month <input type="checkbox"/>	
		(3) NGO <input type="checkbox"/>	(3). 6-11 month <input type="checkbox"/>	
	(4)Apprentice <input type="checkbox"/>	(4).(Specify)		

SECTION B (SELF-EFFICACY):

In this section, please provide your response with respect to the following statements concerning how skilled you are in computer programming. Please indicate your response by making a cross (X) in the boxes provided.

1. I can write syntactically correct C# statements

Very Easy <input type="checkbox"/>	Easy <input type="checkbox"/>	Neutral <input type="checkbox"/>	Difficult <input type="checkbox"/>	Very Difficult <input type="checkbox"/>
------------------------------------	-------------------------------	----------------------------------	------------------------------------	---

2. I understand the language structure of C# programming language.

Very Easy <input type="checkbox"/>	Easy <input type="checkbox"/>	Neutral <input type="checkbox"/>	Difficult <input type="checkbox"/>	Very Difficult <input type="checkbox"/>
------------------------------------	-------------------------------	----------------------------------	------------------------------------	---

3. I can write a long and complex C# program to solve any given problem.

Very Easy <input type="checkbox"/>	Easy <input type="checkbox"/>	Neutral <input type="checkbox"/>	Difficult <input type="checkbox"/>	Very Difficult <input type="checkbox"/>
------------------------------------	-------------------------------	----------------------------------	------------------------------------	---

4. I can relate objects from the problem domain to my C# application.

Very Easy <input type="checkbox"/>	Easy <input type="checkbox"/>	Neutral <input type="checkbox"/>	Difficult <input type="checkbox"/>	Very Difficult <input type="checkbox"/>
------------------------------------	-------------------------------	----------------------------------	------------------------------------	---

5. I could complete a programming project once someone else helped me get started.

Very Easy <input type="checkbox"/>	Easy <input type="checkbox"/>	Neutral <input type="checkbox"/>	Difficult <input type="checkbox"/>	Very Difficult <input type="checkbox"/>
------------------------------------	-------------------------------	----------------------------------	------------------------------------	---

6. I have debugged (correct all the errors) complex C# program that I had written and made it work

Very Easy <input type="checkbox"/>	Easy <input type="checkbox"/>	Neutral <input type="checkbox"/>	Difficult <input type="checkbox"/>	Very Difficult <input type="checkbox"/>
------------------------------------	-------------------------------	----------------------------------	------------------------------------	---

7. I could come up with a suitable strategy for a given programming project in a short time.

Very Easy <input type="checkbox"/>	Easy <input type="checkbox"/>	Neutral <input type="checkbox"/>	Difficult <input type="checkbox"/>	Very Difficult <input type="checkbox"/>
------------------------------------	-------------------------------	----------------------------------	------------------------------------	---

8. I could find ways of overcoming the problem if I got stuck at a point while working on a C# programming project.

Very Easy <input type="checkbox"/>	Easy <input type="checkbox"/>	Neutral <input type="checkbox"/>	Difficult <input type="checkbox"/>	Very Difficult <input type="checkbox"/>
------------------------------------	-------------------------------	----------------------------------	------------------------------------	---

9. I could mentally trace through the execution of a long C# program.

Very Easy <input type="checkbox"/>	Easy <input type="checkbox"/>	Neutral <input type="checkbox"/>	Difficult <input type="checkbox"/>	Very Difficult <input type="checkbox"/>
------------------------------------	-------------------------------	----------------------------------	------------------------------------	---

SECTION C (MENTAL MODEL):

In this section, it involves critical thinking to determine your **mental model** and **ability to maintain consistency in resolving basic computer programing problems**. Cross the appropriate answers with (X)

(1) int a = 10; int b = 20; a = b what is the value of a=?	1. 10
	2. 20
	3. 30
	4. 0

(2) Person a, b; a = new Person ("Jack"); b = new Person ("Tom"); b = a; a = b; what is the value of b &a?	1. Jack, Tom
	2. Tom, Jack
	3. Jack only
	4. Tom only

(3) int a = 5; int b = 3 int c = 7; a = c; b = a; c = b; What is the value of a b c?	1. 5,3,7
	2. 7,3,7
	3. 7,5,7
	4. 7,7,7

(4) Person a, b, c; a = new Person ("Jack"); b = new Person ("Tom"); c = new Person ("Jim"); b = a; a = c; c = b; what is the value of b,a, c?	1. Jim, Tom,
	2. Tom, Jim Jack
	3. Jack, Jim,
	4. Jim, Jack, Tom

SECTION D (SUGGESTED ENHANCEMENT):

Please make suggestions with regards to how you think that learning programming skill may be improved.

Thank You for Your Participation

APPENDIX B: ETHICAL CLEARANCE



4 December 2017

Mr Kelvin Samuel Osaji-Onalo 216076915
School of Management IT and Governance
Westville Campus

Dear Mr Osaji-Onalo

Protocol reference number: **HSS/2126/017M**
Project Title: **Factors that influence students' learning of Introductory Computer Programming**

Full Approval – Expedited Application

In response to your application received 1 November 2017, the Humanities & Social Sciences Research Ethics Committee has considered the abovementioned application and the protocol has been granted **FULL APPROVAL**.

Any alteration/s to the approved research protocol i.e. Questionnaire/Interview Schedule, Informed Consent Form, Title of the Project, Location of the Study, Research Approach and Methods must be reviewed and approved through the amendment /modification prior to its implementation. In case you have further queries, please quote the above reference number.

PLEASE NOTE: Research data should be securely stored in the discipline/department for a period of 5 years.

The ethical clearance certificate is only valid for a period of 3 years from the date of issue. Thereafter Recertification must be applied for on an annual basis.

I take this opportunity of wishing you everything of the best with your study.

Yours faithfully

.....
Dr Shenuka Singh (Chair)
Humanities & Social Sciences Research Ethics Committee

/pm

Cc Supervisors: Dr Sanjay Ranjeeth
Cc Academic Leader Research: Prof Debbie Vigar
Cc School Administrator: Ms Debbie Cunynghame

Humanities & Social Sciences Research Ethics Committee


Dr Shenuka Singh (Chair)


Westville Campus, Govan Mbeki Building

Postal Address: Private Bag X54001, Durban 4000

Telephone: +27 (0) 31 260 3587/8350/4657 Facsimile: +27 (0) 31 260 4609 Email: ujmbao@ukzn.ac.za / snvmanm@ukzn.ac.za / mohunp@ukzn.ac.za

Websites: www.ukzn.ac.za

 1910 - 2010 
100 YEARS OF ACADEMIC EXCELLENCE

Founding Campuses  Edgewood  Howard College  Medical School  Pietermaritzburg  Westville



4 December 2017

Mr Kelvin Samuel Osaji-Onalo 216076915
School of Management IT and Governance
Westville Campus

Dear Mr Osaji-Onalo

Protocol reference number: HSS/2126/017M

New project title: Factors influencing the Learning of Introductory Computer Programming at the Durban University of Technology

Approval Notification – Amendment Application

This letter serves to notify you that your application and request for an amendment received on 14 August 2019 has now been approved as follows:

- Change in Title

Any alterations to the approved research protocol i.e. Questionnaire/Interview Schedule, Informed Consent Form; Title of the Project, Location of the Study must be reviewed and approved through an amendment /modification prior to its implementation. In case you have further queries, please quote the above reference number.

PLEASE NOTE: Research data should be securely stored in the discipline/department for a period of 5 years.

The ethical clearance certificate is only valid for period of 3 years from the date of original issue. Thereafter Recertification must be applied for on an annual basis.

Best wishes for the successful completion of your research protocol.

Yours faithfully

Professor Urmilla Bob
University Dean of Research

/ms

Cc Supervisors: Dr Sanjay Ranjeeth
Cc Academic Leader Research: Professor Isabel Martins
Cc School Administrator: Ms Angela Pearce

Humanities & Social Sciences Research Ethics Committee

Dr Shenuka Singh (Chair)

Westville Campus, Govan Mbeki Building

Postal Address: Private Bag X54001, Durban 4000

Telephone: +27 (0) 31 260 3587/8350/4557 Facsimile: +27 (0) 31 260 4609 Email: ximbap@ukzn.ac.za / snvmanm@ukzn.ac.za / mohung@ukzn.ac.za

Website: www.ukzn.ac.za



Founding Campuses: ■ Edgewood ■ Howard College ■ Medical School ■ Pietermaritzburg ■ Westville

APPENDIX C: RESEARCH APPROVAL LETTER



Date: 27/10/2017

To Whom It May Concern:

PERMISSION TO CONDUCT RESEARCH AS PART OF THE RESEARCH DISSERTATION REQUIREMENT FOR MCOM COURSEWORK DEGREE

Name: OSAJI-ONALO KELVIN SMUEL

Student No: 216076915

Dissertation Topic: Factors that Influence Students' Learning of Introductory Computer Programming at the Durban University of Technology.

It is a requirement for the above mentioned student to undertake a practical research project as part of the research dissertation requirement for M.COM Coursework degree.

Typically this project will be a "practical problem solving" exercise, and necessitate data gathering by questionnaires or interviews.

Your assistance in permitting access to your organisation for purposes of this research is most appreciated. Please be assured that all information gained from the research will be treated with the utmost circumspection. Further, should you wish the result from the dissertation "to be embargoed" for an agreed period of time, this can be arranged. The student will strictly adhere to confidentiality and anonymity.

If permission is granted the UKZN requires this to be in writing on a letterhead and signed by the relevant authority.

Thank you for your assistance in this regard.

Yours sincerely

Supervisor: Mr Sanjay Ranjeeth

School of Management, Information & Governance

Postal Address: Private Bag X54001, Durban 4000, South Africa

Telephone: +27 (0) 31 260 7013 Facsimile: +27 (0) 31 260 7569 Email: muteswahm@ukzn.ac.za

Website: www.ukzn.ac.za



Founding Campuses: Edgewood Howard College Medical School Pietermaritzburg Westville

APPENDIX D: REQUEST FOR PERMISSION TO CONDUCT THE RESEARCH



*Directorate for Research and Postgraduate Support
Durban University of Technology
Tromso Annexe, Steve Biko Campus
P.O. Box 1334, Durban 4000
Tel.: 031-3732576/7
Fax: 031-3732946*

29th August 2017

Mr Osaji-Onalo Kelvin Smuelis
c/o School of Information Systems & Technology
University of Kwa-Zulu Natal

Dear Ms Smuelis

PERMISSION TO CONDUCT RESEARCH AT THE DUT

Your email correspondence in respect of the above refers. I am pleased to inform you that the Institutional Research Committee (IRC) has granted provisional permission for you to conduct your research "Factors that Influence Students' Learning of Introductory Computer Programming at the Durban University of Technology" at the Durban University of Technology.

Kindly note, that the committee requires you to provide proof of full ethical clearance prior to you commencing with your research at the DUT.

The DUT may impose any other condition it deems appropriate in the circumstances having regard to nature and extent of access to and use of information requested.

We would be grateful if a summary of your key research findings can be submitted to the IRC on completion of your studies.

Kindest regards.
Yours sincerely

A handwritten signature in cursive script that reads 'Carin Napier'.

PROF CARIN NAPIER
DIRECTOR (ACTING): RESEARCH AND POSTGRADUATE SUPPORT DIRECTION

APPENDIX E: SIMILARITY REPORT

FACTORS INFLUENCING THE LEARNING OF INTRODUCTORY COMPUTER PROGRAMING AT DUT

ORIGINALITY REPORT

8%

SIMILARITY INDEX

5%

INTERNET SOURCES

5%

PUBLICATIONS

%

STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

< 1%

★ studenttheses.cbs.dk

Internet Source

Exclude quotes On

Exclude matches < 5 words

Exclude bibliography On

APPENDIX F: GENERAL FACTOR

Statement	Number of participants
“The terms used in the programming are confusing ”	6
<i>“programming is sweet and logical but we needed to be taught to be logical too”</i>	22
“programming skill demands constant practice, therefore, it is a time-consuming course”	81
“There are too many things to know at the same time “	98
“Programming is competitive and I hate competition”	110
“I am waiting to hear that programming is off the campus”	163
“Programming requires fast learners who are logical but sometimes I am creative but no logical”	174
“if possible we should be given extra classes/tutorials”	192

APPENDIX G: PARTICIPANTS OPINION OF COMPUTER PROGRAMMING

SELF-EFFICACY

Statement	Number of participants
“I found programming challenging at DUT because I was not taught programming in high school. Now I am struggling”	2
“I am lacking motivation or Interest, commitment, and determination in programming”	10
“I am not prepared enough to face the challenges in the chosen course in programming”	37
“I don’t have any knowledge about computers, so I am struggling to know both computers and programming“	45
“Some of us chose the computer programming course because it is in popular demand”	67
“The lecturers/tutors are not patient enough because they are busy with other stuff”	140
“Since students are always on mobile phones maybe coding programming with mobile apps should be suggested ”	156
“Programming needs consistency and it is difficult for me to maintain that”	200

APPENDIX H: CORRESPONDING STATEMENTS USED FOR CORRELATION

Number of Q	A corresponding statement or question number used for Correlation (See appendix A correlation below)
1	Self-Efficacy_Average
2	Mental_Model
3	Performance
4	Means of Previous Experience (5 Likert Scale Items)
5	Person a, b, c; a = new Person (“Jack”); b = new Person (“Tom”); c = new Person (“Jim”); b = a; a = c; c = b; what is the value of b,a, c?
6	int a = 5; int b = 3 int c = 7; a = c; b = a; c = b; what is the value of a b c?
7	Person a, b; a = new Person (“Jack”); b = new Person (“Tom”); b = a; a = b; what is the value of b and a?
8	int a = 10; int b = 20; a = b what is the value of a=?
9	I can write syntactically correct C# statements
10	I understand the language structure of C# programming language
11	I can write a long and complex C# program to solve any given problem
12	I can relate objects from the problem domain to my C# application.
13	I could complete a programming project once someone else helped me get started
14	I have debugged (correct all the errors) complex C# program that I had written and made it work

15	I could come up with a suitable strategy for a given programming project in a short time
16	I could find ways of overcoming the problem if I got stuck at a point while working on a C# programming project
17	I could mentally trace through the execution of a long C# program

**APPENDIX J: DECLARATION CERTIFICATE OF THE ENGLISH
LANGUAGE EDITING OF THE DISSERTATION.**

ASOKA ENGLISH LANGUAGE EDITING

CELL NO.: 0836507817



**DECLARATION CERTIFICATE OF THE ENGLISH
LANGUAGE EDITING OF THE DISSERTATION**

***FACTORS INFLUENCING FIRST YEAR STUDENT'S LEARNING IN THE
INTRODUCTORY COMPUTER PROGRAMMING LANGUAGE COURSE AT DUT***

Candidate: **Osaji-Onalo KS**



DISCLAIMER

Whilst the English language editor has used electronic track changes to facilitate corrections and has inserted comments and queries in a right-hand column, the responsibility for effecting changes in the final, submitted document, remains the responsibility of the candidate in consultation with the supervisor/promoter.

Director: Prof. Dennis Schaffer, M.A.(Leeds), PhD, KwaZulu (Natal), TEFL(London), TITC Business English,
Emeritus Professor UKZN. Univ. Cambridge Accreditation: IGCSE Drama. Hon. Research Fellow, DUT.
Durban University of Technology.