Developing a web based blended learning technique to improve computer programming competence of information technology students

Submitted in fulfilment of the requirements
of the Degree of
Master of Information & Communications Technology
in the Faculty of Accounting and Informatics at
Durban University of Technology

By

Priyalushinee Jackson

February 2017

Supervisor: Professor O.O Olugbara

DECLARATION

I, Priyalushinee Jackson, hereby declare that the content within this dissertation is my own work. All sources that I have used or quoted have been acknowledged in the text by the means of completed references. This study has not been previously submitted in any form to the Durban University of Technology or to any other institution for assessment or for any other purpose.

_____ _____

Priyalushinee Jackson Date

Approved for final submission

_____ _____

Professor O.O. Olugbara PhD Date

ACKNOWLEDGEMENTS

First and above all, I praise God for providing me this opportunity and granting me the strength, capability and courage to complete my studies successfully. This dissertation would not have been possible without the guidance and help of several individuals who assisted in the preparation and completion of this study. I would therefore like to offer my sincere thanks to all of them. Firstly, my sincere thanks and deep gratitude goes to my supervisor, Professor Oludayo, O. Olugbara. His diverse knowledge, constructive comments and valuable insight about this research area has added great value to my research dissertation. Without his understanding, patience, and continuous support this piece of work would never have been completed.

I am forever thankful to all my colleagues at the Department of Information Technology for their continued supported sound advice. I want to acknowledge my inspirational parents, I am forever thankful to my parents who are my God's, for they have supported me even when I was experiencing difficulty while studying. They gave me the courage and will to continue. Without their wise words and kind gestures, I would not have been able to complete my studies. Mrs Rani Perumal, thank you for your support and encouragement and being my pillar of strength. I also want to acknowledge my loving and caring dad, Mr Reuben Perumal who assisted me with the kids whenever needed, he was also guiding me and ensuring I complete this study. I want to thank my beloved husband Denzel Jackson, whose endless support from start to finish encouraged me to complete this work. My deepest love to my beloved daughter who suffered a lot during this journey and complained very little. To my little boy I am thankful god has sent you here to be with me. I am so happy and excited to have reach this milestone in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

ABSTRACT

Computer-programming dexterity is an essential skill for students of computer science, information technology and engineering who are intrinsically expected to be able to do programming. However, teaching and learning computer-programming concepts and skills have been recognised as a great challenge for both teachers and students, for many reasons. Computer-programming requires new ideas in thinking and conceptualising practical solutions. It requires creative skills in solving practical, but often difficult problems. Moreover, computer-programming students, generally lack problem-solving skills and self-efficacy. They typically find it difficult to use artificial programming languages to solve challenging problems. There is the problem of poor background in science and mathematics that would help students to rapidly understand the intricacies of computer-programming. Students are not motivated to overcome the fear of the bizarre syntax of computer-programming codes. These challenges, coupled with the huge potential of computing applications in the society have made the development of effective pedagogies and environments for computer-programming courses, an important issue. To address this issue in a unique way, this study proposes to explore a web-based, blended learning technique with minimal instructor intervention, to improve the computer-programming competence of information technology students. These students are expected to have developed an acceptable level of computer-programming competence at university to be job ready and to be self-reliant in their future careers. The technique being proposed in this study was implemented in a blackboard ™/®/© web-based environment. The effectiveness of the technique was demonstrated using experimentation coupled with the data analysis method that is based on the three-parameter item response theory and retrospective pretest. The method used in this study to evaluate computer-programming competence of students reflects the perspective of the students in the evaluation process. The results of the study indeed show that using the proposed technique, information technology students dynamically collaborate with their peers with minimal instructor intervention towards improving their computer-programming competence.

CHAPTER ONE: INTRODUCTION

This dissertation reports on the development of a web-based blended learning technique to improve computer-programming competence of information technology students at the Durban University of Technology (DUT). The drop-out and failure rates of information technology students are very high at DUT, as at many other universities (Bennedsen and Caspersen 2007; Mohammed and Mohan 2010; Ranjeeth and Naidoo 2011). Students experiencing enormous computer-programming problems are from the information technology and computer systems departments at the University. Although they are studying computer-programming modules at DUT, they share the same computer-programming problems as many other students across the globe (Bennedsen and Caspersen 2007; Ranjeeth and Naidoo 2011). The computer-programming problems often experienced by university students have been identified by many authors, who note that computer-programming competence was not well-developed in students (Ranjeeth and Naidoo 2011; Hsieh, Lee and Su 2013; Isong 2014). Students appear to lack the background knowledge of important concepts in computer-programming courses, and are unable to form new knowledge without prior knowledge (Campbell 2008; Rane-Sharma *et al.* 2010; Boughey 2013).

As we all know, information communication technology (ICT) has changed our planet into an society focused on innovation (Danner and Pessu 2013), where one of the prime motivations is acquiring computer-programming, information literacy, multimedia literacy and communication competences. These competences are essential for a student to be able to work efficiently in a society predominantly driven by ICT (Danner and Pessu 2013). Therefore, we need to take the necessary steps to ensure that our students can handle a tertiary environment that prepares them adequately to obtain the necessary competences that would cause them to be successful at higher education institutions, and in their future careers. Technology in the 21st century and its diverse approaches have dominated our world as we know it. Almost every task performed in offices today involves some kind of interaction in a computerised environment (Shannon and Bennett 2012; Atif 2013; Barik, Jena and Sethy 2014). As a result, university students studying computer-programming are faced with the enormous challenges of being able to demonstrate their programming skills in society. Hsieh, Lee and Su (2013) have found evidence to support a change at universities from utilising the conventional teaching and learning approaches to

technology oriented approaches that make use of the web-based systems to enhance their teaching and learning of the computer-programming courses. The staff at DUT wanted to find out the trend of new approaches to teaching and learning computer-programming and had implemented a web-based blended learning pedagogy into the curriculum of the information technology courses. The web-based blended learning pedagogy has been implemented in the Programming 1, Programming 2 and Computer systems courses, with a view to improve the computer-programming competence of these students using a different methodology previously used.

Rosli, Ibrahim Teo and Khairol Azmi (2010) had found evidence that students find problem-solving and writing of computer programs an exceptionally challenging aptitude to master, because students have difficulty understanding the prerequisite knowledge and producing a working prototype system. Previous researchers have also found that students have no basic knowledge about computer systems, had not been exposed to any computer orientation and their problem-solving skills were not adequately developed (Chen, Su and Liu 2007; Cha *et al.* 2011; Shannon and Bennett 2012; Isong 2014). In particular, first year students wanted to study information technology at the university, but they have no prior knowledge of computers as a foundation on which to build new knowledge. Since there is no prior knowledge, they find that problem-solving is boring and cumbersome, which leads a student to abandon their studies. Mathews (2010) also found substantial proof to support the theory that computer and analytical programming skills are very important to obtain and use, because these require students to solve real-life, challenging problems. These programming skills are relevant in order for a student to analyse and solve diverse, but challenging problems, while also developing their computer-programming competence. Computer-programming competence is used as the basis for creating innovative and creative computerised systems, and has emerged as a crucial issue at many universities (Wang *et al.* 2011; Schumm *et al.* 2012; Atif 2013; Altintas, Gunes and Sayan 2014; Brito and de Sá-Soares 2014; Isong 2014; Yang *et al.* 2015). Consequently, since many researchers echo the same evidence about students finding problem-solving very challenging where there is no adequate prior knowledge, dropout and failure rates are reportedly very high (Bennedsen and Caspersen 2007; Mohammed and Mohan 2010; Rosli, Ibrahim Teo and Khairol Azmi 2010).

Many universities have adopted the traditional approach to teaching and learning. However, Isong (2014), Berry and Kölling (2013) found that students were experiencing challenges while

learning computer-programming at higher education institutions using the traditional approach. Consequently, this study uses a unique pedagogy for teaching and learning of computer-programming skills in an attempt to improve the computer-programming competence of information technology students. Retrospective pretest experiments were performed in this study to determine whether the implemented web-based blended learning pedagogy (Mathews 2010) had improved computer-programming competence of information technology students. There are many research papers and dissertations that have contributed to either using a web-based or traditional pedagogy in higher education institutions. However, this research dissertation had used a unique technique based on the hybrid of two respective techniques to develop computer-programming competence of information technology students. There are many advantages of using this unique approach to teaching and learning, when compared it to the conventional approach that many higher education institutions across the world have been adopting.

Mathews (2010) has documented the success of adopting a studio based pedagogy in his course at Wisconsin University. This benefitted the students immensely, because work and interaction were effectively executed within their groups. They engaged actively with other students in groups using all their prior knowledge and understanding to develop their competence further. The author used an intervention with minimal instructor supervision and it was successful in developing students' competencies. However, the current study had incorporated the blended learning pedagogy, together with the blackboard online classroom, to support the needs of all information technology students. There were enough resources available and everyone was able to communicate using online group discussions and peer interaction. A web-based blackboard environment was used to assist in the training of community development workers in the Kwazulu-Natal province to improve their e-skills. This study proved to be a success as the community development workers had responded positively towards the e–skills training conducted with the aid of the blackboard system (Olugbara *et al.* 2014). This is due to the fact that the above authors used the blackboard online system, and it had facilitated the development of e-skills. Therefore this study used the online classroom in the blackboard learning management system, together with the blended learning pedagogy, to implement a unique technique that would be able to change the teaching and learning approaches at DUT. The university is currently driving its e-learning strategic objective using the blackboard system as the learning management environment of choice.

1.1.Research Problem

Learning computer-programming is a complex activity, because computer-programming requires conceptualising new ideas in thinking and creative skills in problem-solving (Emmanuel, Oluwadamilare and Yomi 2015). In particular, developing the computer-programming competence of students who are entering the university for the first time is a daunting task. There are many factors that can be attributed to this challenge, amongst which are: lack of problem-solving skills; lack of self-efficacy; difficulty using artificial languages different from natural languages; poor science and mathematics background as well as motivation; class size; bizarre programming language syntax; and inability to apply programming constructs to solve practical problems (Hooshyar *et al.* 2015). In particular, authors have found that linking prior knowledge with current knowledge is difficult, especially if students have no prior knowledge as the case may be in computer-programming courses (Campbell 2008). Students cannot formulate a connection that exists among the scheme of work and the methods of assessment (Rust 2002). This gap was identified by Corney, Teague and Thomas (2010), Rane-Sharma *et al.* (2010) and Boughey (2013), who explain that the lack of understanding of simple computer-programming concepts affects the competence of students to solve challenging problems. This could in turn have an adverse impact on the comprehension of more difficult knowledge delivered to students as they progress to higher levels at universities. Students are "under-prepared" for real university education, because the traditional pedagogy of teaching and learning does not cater for collaboration and active engagement in the learning process (Isong 2014). Consequently, one could assume that finding a better pedagogy for teaching computer-programming to students would likely improve their computer-programming competence. The basic assumption of this study is that engaging peers is very significant for developing their computer-programming competence, because their skills are strengthened by interaction (Claro *et al.* 2012). This study has contributed to scientific knowledge by answering the following important research question:

How can a blended learning pedagogical technique be implemented in a web based teaching and learning environment to practically improve the computer-programming competence of information technology students?

1.2.Research aim and objectives

The overarching aim of this study is to develop a blended learning pedagogical technique in a web-based teaching and learning environment that can practically improve the computer-programming competence of information technology students. In order to accomplish this research aim, the following objectives are set.

a) To explore how a blended learning pedagogical technique could be implemented in a web-based teaching and learning environment that could help students in designing, understanding, communicating and collaborating to improve their computer-programming competence.
b) To implement a blended learning pedagogical technique in a blackboard web-based teaching and learning environment to support the development of computer-programming competence of information technology students.
c) To determine whether a blended learning pedagogical technique implemented in a blackboard web-based teaching and learning environment improves the computer-programming competence of information technology students.

1.3. Significance of the study

The topic of developing computer-programming competence of students is an important one. The development of effective learning strategies and environments for programming courses has become an important issue (Yang *et al.* 2015). The rapid development of advances in information technology and their direct applications to the society has created high demand for skilful programming specialists. As a result, acquisition of programming skills has become a core competence in computer science, information technology and engineering. Students from these interrelated fields are required to be able to do programming, as there are several programming courses in these fields that require programming competence (Hooshyar *et al.* 2015). In general, improving the academic performance of students is part of the continuity plan to uplift the standard of education in the areas of computer science, information technology and computing related disciplines. The significance of this study is to determine how a web-based blended learning pedagogy could help to improve computer-programming competence of students from information technology and computer systems.

1.4. Contribution of the study

This research study has contributed significantly to developing computer-programming competence. Previous research has often mentioned that students are finding it difficult to develop critical skills, as exemplified by computer-programming competence (Ming-der Wu 2012). Improving computer-programming competence of computing students has not been sufficiently dealt with in the literature. This study has demonstrated that a seamless implementation of blended learning pedagogy in a blackboard web-based teaching and learning environment has the potential to improve computer-programming competence of information technology students. The work of Hooshyar *et al.* (2015), using a flowchart-based programming environment to improve problem-solving for minority students in computer-programming, supports the need for an alternative. The missing gap in their environment, which this study considers to be germane, is the absence of active learning through free collaboration among students.

To date, a considerable body of research has sought to develop different strategies for improving computer-programming competency of students. More recent on the list are the flowchart-based programming environment (Hooshyar *et al.* 2015); recommended strategies (Emmanuel, Oluwadamilare and Yomi 2015); a two-tier test-based approach in a web-based learning environment (Yang *et al.* 2015); and mobile games approach (Barchino *et al.* 2012). While this research provides a number of important insights, especially in regard to strategies, recommendations and environment, they have not explored how the assessment of those approaches improves computer-programming competence from the student's perspective. The present work is designed to be the first to apply the methods of retrospective pretest and item response theory to determine that a blended learning approach implemented in a blackboard web environment has indeed improved computer-programming competence of information technology students. In addition, this research seeks to fill the gap on how student competence is traditionally assessed from a teacher perspective, while ignoring a student perspective. Hence this research has introduced the idea that competence, skill and knowledge of students can be better assessed from their perspective, a concept relevant to student-centred learning.

1.5.Chapter synopsis

Chapter 1 of this dissertation introduces the problem investigated, followed by the aim and objectives of the research, thereafter followed by the significance of the study and lastly discusses the contributions of this study.

Chapter 2 provides a comprehensive overview of the existing literature on web-based blended learning techniques used to date. In addition, it provides the literature that explains the theoretical foundation for the research methodology and the research objectives.

Chapter 3 presents a detailed account of all the steps of the methodology carried out to achieve the objectives of this study, including the selection of research techniques to solve the research question posed.

Chapter 4 presents the results of the questionnaire administered in this study and the results of assessing the computer-programming competence of information technology students, which is discussed in detail.

Chapter 5 provides a summary of what the research had achieved, states the conclusion of the research, and identifies future research ideas.

CHAPTER TWO: LITERATURE REVIEW

This chapter highlights the main literature concepts related to this study, providing an insight into the research domain. It starts by enunciating the challenges of developing computer-programming competence, followed by defining the main concepts of competence theory, how prior knowledge can assist in developing computer-programming competence, the different methods for improving computer-programming competence, and how these concepts relate to improving computer-programming competence. The challenges of developing computer-programming competence provide the basis for the understanding of this competence. Thereafter, the relationship between prior knowledge and competence development is discussed. Crucial to this study is the assessment of computer-programming competence, which forms part of the literature review. Thereafter, a detailed discussion of factors influencing computer-programming competence paves the way for the discussion of methods to improve it.

2.1 Challenges of Developing Computer-programming Competence

There is a general consensus in the literature that students encountered challenges when trying to solve computing problems. Many of the students often fail to go through the necessary steps to ensure that they had a deep understanding of the programming concepts, to be confident enough to find the right algorithmic solution to a particular problem and therefore their computer-programming competence is not well-developed (Altintas, Gunes and Sayan 2014). This has hindered the competence of students to learn new knowledge in the computer-programming courses. It is very difficult to help students develop their computer-programming competence, when they have little or no prior knowledge of computers or programming. The computer-programming course offered at DUT, uses the object-oriented approach, where lecturers use real world computing problems in the classroom. A scheme of work was designed to cover the basic concepts of computer-programming in the first few weeks of the semester. The more advanced concepts are then taught during the remainder of the semester. However, students are still trying to grasp the basic concepts of computer-programming. Therefore the more advance concepts are introduced and tested in the form of an examination. Consequently, students are unable to gain an in-depth understanding of the more challenging concepts, which are based on their prior knowledge of computer-programming. Students are facing a number of challenges because they have not understood the basic concepts properly. Therefore they cannot grasp the more advanced

concepts. There are many challenges in teaching and learning of computer-programming because students need a higher reasoning capability to solve difficult problems (Govender *et al.* 2013). Computer-programming knowledge has been a field that students had limited knowledge and understanding, while using the traditional approach (Govender *et al.* 2013).

The understanding of the core challenges of developing computer-programming competence can be useful for curriculum designers and educators to ensure that new information technology and computer science students can make a smooth transition into their new discipline. The challenges of developing computer-programming competence have led to relatively high dropout and failure rates in the first year introductory programming courses that include the following (Butler and Morgan 2007; Connolly, Murphy and Moore 2009; Agarwal and Hundhausen 2010; Mohammed and Mohan 2010; Ming-der Wu 2012; Danner and Pessu 2013; Mhashi and Alakeel 2013; Altintas, Gunes and Sayan 2014; Isong 2014; Reardon and Tangney 2014), where:

a) the conceptual difficulty of the various elements of the curriculum that require abstract and logical thinking;
b) the low level of feedback that is available to students with regard to various components of the programming assignment;
c) the poor performance in computer-programming is exacerbated by lack of understanding, hardworking and independent study patterns of the students;
d) the students lack the competence to conceive, design and solve practical problems at its different stages and the general programming topics;
e) the majority of the students lacks the skills to analyse a short piece of computer-programming code;
f) the poorly designed course contents and deficiency of students English Language comprehension introduces a bottleneck;
g) there is a lack of sufficient programming skills by lecturers in organising the required learning materials; and
h) there is a failure by the university to provide the adequate training resources and learning environment that could support programming competence development.

2.2 Competence Theory for Competence Development

Global competence is simplified into three categories, which are: information, point of view, and capacity. It refers to the capability of an individual to engage with others to outline different approaches to everyday challenges (Cutler and Borrego 2010). Competence has been defined as the competence of an individual to comprehend knowledge and complete a specific task (Ming-der Wu 2012). Competence can also be viewed as the strength of a person who is motivated to understand knowledge and successfully apply it. The European Commission outlined competence as the grouping together of comprehension, skilfulness and attitudes that an individual might obtain in a given situation (Ilahi, Belcadhi and Braham 2013).

The competence motivation theory states that competence motivation increases when a person skilfully masters a particular task (Burgess, Grogan and Burwitz 2006). The competence to master a task encourages the person to master more tasks (Burgess, Grogan and Burwitz 2006). Motivation ensures that some tasks get completed, where a student requires the competence to be motivated enough to succeed, and ensure he had developed computer-programming competence (Serrano-Cámara *et al.* 2014). When a person finds writing computer programs interesting and acceptable to his learning needs and styles, then learning to develop competence can follow (Fernandez *et al.* 2015). The adapted Kirkpatrick's model illustrates the need for strong motivation to learn, followed by a positive experience in a learning event, which itself increases the motivation to learn and develop competence (Fernandez *et al.* 2015).

Competence has been found to be directly linked to self-efficacy and academic performance. Fessakis, Gouli and Mavroudi (2013) found that computer-programming competence enhances higher request thinking and creates critical thinking abilities, which have assisted students in practical solve computing problems. Therefore computer-programming competence has emerged as a very scarce skill that students need to develop. Self-efficacy refers to the competence of someone to do a given task, plus the degree of confidence one has in the task (Malliari, Korobili and Togia 2012). The Harters competence motivation theory is linked to competence and self-efficacy, which explains that a person's competence motivation increases when they master a skill (Burgess, Grogan and Burwitz 2006). However, the level of confidence is very important, and can control the depth of knowledge learnt. Every graduate of information technology and computer systems should have acquired computer-programming competence, because of the

continued dependence on technology in the modern society of today (Malliari, Korobili and Togia 2012).

2.3 Prior Knowledge and Critical Thinking for Competence Development

2.3.1 Prior knowledge

Prior knowledge is directly linked to competence development, because competence development requires one of the key elements in acquiring new knowledge and skills, which is prior knowledge. Prior knowledge was linked to learning new knowledge, which had helped students to develop their competence to learn (Judson 2012). Students constructed, developed and demonstrated new knowledge using their prior knowledge (Svinicki 1993; Cordova *et al.* 2014). Consequently, the foundation information is based on the crude material that conditions new learning, because prior knowledge is linked to students performance in academia (Campbell 2008). Previous research showed that students were encountering the same challenges of understanding new knowledge, because they did not have any prior knowledge (Svinicki 1993). There are many ways to motivate students to obtain new knowledge, dependent on comprehension of previous understanding in a teaching and learning setting (Judson 2012; Cordova *et al.* 2014; Taub *et al.* 2014).

Similarly, lecturers tested students prior knowledge before starting to teach new concepts, which gives lecturers a sense of each student's prior knowledge (Svinicki 1993; Cordova *et al.* 2014). Svinicki (1993) thus discovered that students found new subject knowledge simple to grasp, and less difficult to comprehend, because their prior knowledge was already developed. Prior knowledge is an important factor to assist students in the acquisition of new knowledge, and it is possible to use it to identify a different method for self-managed learning (Chen and Huang 2013; Taub *et al.* 2014). Cordova et al. (2014) found interpreting and understanding course information previously learnt to be a significant aspect crucial to the success of learning, and discovered that a conventional method of comprehending prior course information and linking it to existing content was displayed in different courses (Cordova *et al.* 2014). After determining that prior knowledge is precise and correct, it is possible that learning occurs because students combined prior and new knowledge (Cordova *et al.* 2014).

Computer-programming skills have become a core competence for engineering, information technology and computer science students (Yang *et al.* 2015). Students who study computer-programming courses have to understand prior knowledge of the computer-programming dialect, and know how to construct and develop computer codes to prove that they have gained problem-solving skills (Yang *et al.* 2015). Moreover, lecturers of computer-programming found that the uniqueness of this course is found specifically in the way in which it bases the broadening of student knowledge of new concepts on what they already understand. Lecturers need to be able to fluently communicate to students what is required to be taught in class, so that all students reach the same learning level and can comprehend new knowledge and skills (Campbell 2008). Campbell (2008) utilised a classroom approach that used prior understanding of contents as a base for the process of activation of prior understanding, before continuing to teach new knowledge to students.

There are two types of knowledge that Campbell (2008) and Judson (2012) have identified. The first type of knowledge involves comprehension of prior knowledge and abilities. The second type of knowledge identified students that are not able to develop computer-programming competence, because they had no prior knowledge of computers and the programming content (Campbell 2008; Judson 2012). Many authors have agreed that new knowledge cannot be created before a student has gained prior knowledge. In an experimentation to investigate the relationship between competence development and prior knowledge, two groups of pre-service and in-service teachers were studied. The pre-service teachers had prior knowledge so they were driven and determined to complete all the tasks in the course. However, the in-service teaching staff had no prior knowledge about the course (Zottmann *et al.* 2013). Therefore, some of the participants were demotivated, and opted to drop out (Zottmann *et al.* 2013). Since this had occurred with teaching staff, students experienced the same challenges, and chose to dropout. When students have prior knowledge of a subject, it becomes easier to understand new knowledge. However, if there is no prior knowledge, students feel demoralised, demotivated and they perceive the course as difficult, which may eventually lead them to drop out.

A new approach to students gaining knowledge using a self-managed learning technique that encompassed students thinking, designing and interacting to acquire the skills and knowledge in a subject (Taub *et al.* 2014). A game-based learning framework (Chen and Huang 2013) was used to compare two games, the first game called CSI that had positively influenced students as they gained revelatory information (Chen and Huang 2013). However, the game, called

Machinarium, negatively influenced students because of the different types of procedural knowledge obtained (Chen and Huang 2013). Students were shown to have high learning abilities, because their prior knowledge was developed for both games, therefore the "nature of knowledge" determined the positive or negative effects that the game-based learning produced (Chen and Huang 2013). In another study, Altintas, Gunes and Sayan (2014) found that students did not gain computer-programming competence at university within the first six months of studying a computer-programming course, which affected their competence (Law, Lee and Yu 2010). Since they had no prior knowledge, they could not develop their computer-programming competence.

2.3.2 Critical Thinking

Critical thinking, ability and disposition to use this skill are important factors that influence the computer-programming competence of an individual. Critical thinking is defined by Shannon and Bennett (2012) as sensible and intelligent imagining that is centred around choosing what to trust or do. It is a skill acquired based on the level of inquiry from an individual. The authors looked at critical thinking and cognitive learning skills, which must be developed, as factors influencing the computer-programming competence of students. This is because creating, listing, summarising, analysing and assessing have some resemblance to the skill categories in the cognitive learning approach of Blooms taxonomy (Forehand 2010). Carter and Hundhausen (2011) found that the critical thinking skills of students during the review quality process was very weak at the start of the course, however, it improved over the semester, because of the peer-learning techniques that were implemented for this course.

2.4 Factors Influencing Competence Development

Students studying at higher education institutions around the world have found that their computer-programming competence was not well-developed. This had led to their poor understanding of computer-programming concepts and their application to solve challenging problems (Rosli, Ibrahim Teo and Khairol Azmi 2010; Berry and Kölling 2013; Brito and de Sá-Soares 2014; Yang *et al.* 2015). The challenges of not understanding computer-programming is highly prevalent among university students (Hare 2013; Vivian, Falkner and Falkner 2013; Brito and de Sá-Soares 2014; Isong 2014; Yang *et al.* 2015). Hence, they choose to give up, fail or drop out of the computer-programming course (Mohammed and Mohan 2010; Rosli, Ibrahim

Teo and Khairol Azmi 2010; Hare 2013; Reardon and Tangney 2014). When students find that they are repeatedly failing, and cannot concentrate on a particular computer-programming concept, the idea of continuing to learn seems very boring and unappealing to them (Law, Lee and Yu 2010; Hsieh, Lee and Su 2013). This affects their self-esteem because they feel demotivated and they withdraw from the module, therefore, the module is incomplete (Law, Lee and Yu 2010; Hsieh, Lee and Su 2013). Information technology and computer science students in particular were not able to develop their computer-programming competence (Reardon and Tangney 2014). This is a critical skill that must be acquired based on students having the necessary prior knowledge of basic computer-programming concepts, which can make the creation of new knowledge easier to comprehend (Cordova *et al.* 2014). The need to enhance the learning of computer-programming influences students learning inspiration as a capability and computational intuition, which is a key aptitude in the specialised society of today (Malliari, Korobili and Togia 2012).

Knowledge and inspiration are important aspects of gaining knowledge at universities, because knowledge and inspiration are linked to the actions of students (Ngan and Law 2015). Inspiration was identified as the tendency of a person to insist on an exertion, which guides them to achieving a given target (Ngan and Law 2015). The "intrinsic" factors of inspiration are participating or performing for its own sake, as opposed to doing so in order to obtain something external. However, extrinsic factors happened when people are roused to perform a specific conduct or take part in a movement to gain a prize (Ngan and Law 2015). Consequently, inspiration is an important factor that influences the development of computer-programming competence, and it plays an essential role in a student's success. Experience has shown that many students who find themselves in a computer-programming class often have a different notion of what the course requirements are, which ultimately affects their competence. At the university where this study was conducted, some students, who have not done well in the computer-programming class, have mentioned that they felt information technology to be limited to the use of computers, but were surprised when they were required to develop computer codes. These students often performed poorly in the class, because their knowledge was limited and inspiration levels were very low.

Shannon and Bennett (2012) found evidence that suggests students did not understand comprehension of previous patterns, therefore they could not recall these patterns. This comprehension and recalling of information was another factor that might influence the

computer-programming competence of students because students did not comprehend prior knowledge and they could not build new knowledge (Shannon and Bennett 2012). Therefore, Rane-Sharma *et al.* (2010) explained that there are many pedagogical and technological factors included when teaching and learning a computer-programming module. Students need to have developed some basic computer-programming competence to ensure that the goal of obtaining new skills at universities is met; otherwise they are not able to critically think and solve real-life problems.

Students need to have analytical programming skills, which encompass practising long sessions on different real-world problems. If this skill was not developed in students, the outcomes are going to be very bad, as students would have weak problem-solving abilities and this could lead to high failure rates (Mohammed and Mohan 2010). Self-efficacy is another component for developing computer-programming competence, as it is directly linked to academic success. Self-efficacy is one of the important factors for competence development of programming skills. However, self-efficacy will not be pursued for this research, because the target respondents are primarily first year students who are tangent. This kind of student is generally regarded as undeveloped, since they have not developed self-efficacy, as the majority were exposed to programming for the first time.

Finally, it is important to conclude this section by stating that socioeconomic factors generally influence computer-programming competence. Students increasingly bring complex social issues to the university, and they choose to manage their educational challenges in their personal lives in a variety of ways. Socioeconomic statuses, such as funding and emotional maturity are at-risk factors that can inhibit the development of the computer-programming competence of students. For example, having a computer at home can have a positive effect on the development of computer-programming competence of a student.

2.5 Measurement of Competence Development

The measurement of competence is an area in education where significant progress has been made, and it is of particular importance to researchers (Srikant and Aggarwal 2014). Competence measurement forms a very important role in educational institutions, because it tracks the competence of students (Savignon 1976; Ilahi, Belcadhi and Braham 2013). In addition,

competence measurement inspires the students to study harder and it enhances their learning (Kuo and Wu 2013). The measurement of computer-programming competence is of particular importance as it is a topic of concern for many researchers (Srikant and Aggarwal 2014). Web-based measurement models have become increasingly popular among tertiary institutions, and has been included into the curriculum to test the competence levels of students (Ilahi, Belcadhi and Braham 2013). Moreover, it gives the lecturers an idea of the capability of individual students and what they have learnt in the study (Savignon 1976). The only way that students display the understanding of a particular knowledge is to write a test or an examination for each course, which results in either a pass or a fail grade (Brown 2004).

2.5.1 Miller's Competence Pyramid

The Miller's competence pyramid shown in Figure 2.1 is used by the author as a convenient framework for competence measurement (Miller 1990). The Miller's competence pyramid is composed of four levels of knowledge constructs of "knows", "knows how", "shows how" and "does". This pyramid discriminates between four knowledge constructs and provides mechanisms for competence measurement. At the lowest level of the Miller's pyramid is the knowledge type called "know that" or simply "knowledge", which is a formulation that is often used to describe the knowledge of knowing things or reciting a relevant theory. In the context of this study, it refers to the measurement of what students of information technology and computer studies know about computer-programming. Forming the base of the Miller's pyramid represents the foundation knowledge upon which computer-programming competence is built. The measurement of knowledge should consist of test or examination questions that focus on what students know about computer-programming.

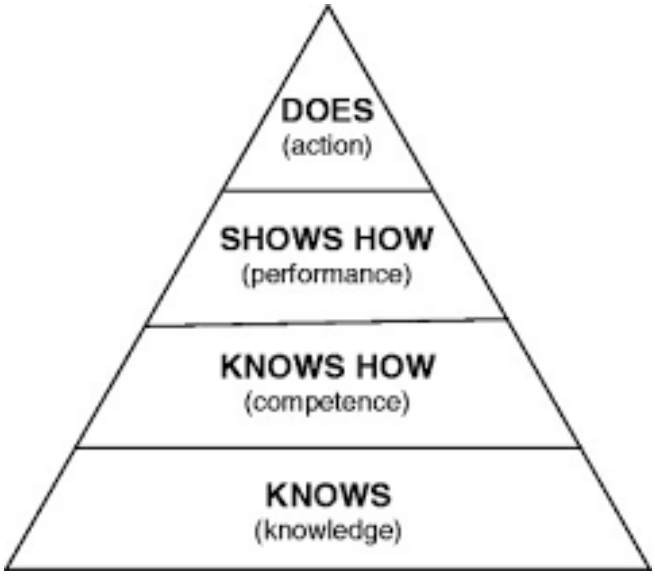

Figure 2.1. – Miller's pyramid ((Miller 1990))

The Miller's pyramid provides another knowledge type referred to as "knows how" or comprehension. This knowledge type can be equated to the knowledge base required for effective functioning of students as computer programmers. Whilst this knowledge base is necessary, it is insufficient. Comprehension in Ryle's formulation describes the capacity to perform a function, and it is distinct from describing the area of knowledge related to the reciting of the relevant theory. In the context of this study, it is the functional knowledge that emphasises the competence of students of information technology and computer studies to comprehend how to program a computer using the necessary tools, but not to articulate a description of what they know about computer-programming. This type of knowledge does not preclude one's skills to articulate what is known, but merely emphasises the skills to perform an act such as programming the computer. The distinctive feature of "know how" and "know that" knowledge types is that the knower's orientation is concerned with performing a function such as programming the computer, rather than describing certain concepts of computer-programming.

The "shows how" or skill is the Miller's knowledge measurement of how a user is able to integrate "knows that" and "know how" into a successful outcome with the implementation of computer systems. Although students may "know that" and they may "know how", they may not be able to integrate knowledge or skill into a successful outcome expected. The emphasis on the acquisition of skills constitutes the competence of the students to choose and perform some computer-programming activities or actions in an appropriate and effective manner, such as developing efficient computer codes to solve a particular problem. The only measurement of

skill is to set a practical test such as placing a student before the computer, using which, the student must write computer codes. This knowledge type contrasts with comprehension, where the emphasis is on the competence to perform an action, but skills are usually associated with some judgement foundation knowledge, concerning such matters as for example when one would perform the action. In this particular case, students of information technology and computer studies should be able to select the right tools to solve a practical computing problem.

The highest level of the Miller's pyramid is the knowledge type referred to as "does", which focuses on methods that provide an assessment of routine competence. This knowledge type involves the translation of skills into actions for the purpose of competence improvement, for example the actual use of a computer system by students to demonstrate how such a system improves performance in their daily business. The measurement method at the "does" level is characterised by reliance on information from knowledgeable people such as a panel of judges to judge competence (Van der Vleuten *et al.* 2010).

2.5.2 Competence Test and Self Reporting Measurements

The measurement of competence or learning has been one of the core functionalities of the education system over the last decade. The competence measurement methods are usually based on the stimulus-response format, such as those used in semester tests and final examination. It has been suggested that what is being measured is determined more by the format of the stimulus than by the format response (Van der Vleuten *et al.* 2010). However, there are many problems often encountered during the creation of the measurement response procedures (Carless 2014). The traditional methods for competence measurement posed a challenge because students were not driven and interested in learning, which has led to the use of other methods to improve student learning (Brown 2004). The studies that used hypermedia together with computerised instructional devices were used for different instructive settings, including educator preparing (Zottmann *et al.* 2013). Computerised systems for learning empowered students to work with audio-visual files in creative routes (Zottmann *et al.* 2013). The measurement of competence ought to encompass both formative and summative assessments (Carless 2014). Student-centric competence measurements were based on the learning that was facilitated during the course (Carless 2014). Brown (2004) had argued that measurement of learning should be done in a student-oriented environment, where tests should be created using the "evidence of

achievement" rather than the evidence of memorising and recalling of information. Moreover, students demonstrated the concepts of learning a topic by applying this knowledge to achieve the desired learning outcome (Brown 2004). The adjustment to the type of examinations that students are assessed by is highly likely to render a change in the knowledge of students.

Competence-based measurement models involve three interconnected aspects, "subject matter, capability and context" (Ilahi, Belcadhi and Braham 2013). However, "the intended learning outcomes and contexts" are that which ought to constitute the measurement of skills (Ilahi, Belcadhi and Braham 2013). In addition, the "intended learning outcome" comprised of the competence and topic (Ilahi, Belcadhi and Braham 2013). Skill-based measurement is a prototype that was created from an arrangement of results (Ilahi, Belcadhi and Braham 2013). The process of obtaining data about the skill set of a person was the purpose of a competence-based measurement. However, the online competence based assessment model that was proposed by Ilahi, Belcadhi and Braham (2013) was testing an online instrument and prototype created to help a student gain a specific skill. A student should achieve an acceptable level of competence, based on the outcomes for a specific context. Consequently, competence is subject-specific, and non-generic (Van der Vleuten *et al.* 2010), so one cannot measure competence using generic knowledge elements.

Competence tests and self-reporting are two important methods for measuring competence development. The general performance of competence tests has been compared with self-reporting by students, to measure their competence level. There is no general consensus in the literature about whether there is a correlation between these two methods. Most studies conclude that there is no correlation between how people self-report their level of competence and how they perform in tests (Van Vliet, Kletke and Chakraborty 1994; Kreth *et al.* 2005; Ballantine, Larres and Oyelere 2007). Low-performing persons tend to overestimate their capabilities, whilst there are studies which have found a correspondence between self-reporting and test results (Hakkarainen *et al.* 2000). However, self-reporting used in isolation can be generally unreliable and misleading.

2.5.3 Retrospective Pretest Competence Measurement

The methods of retrospective pretest and conventional pretest and posttest have proven to be successful in measuring competence during a small timeframe. The retrospective pretest measurement can be conducted once (after the intervention) as opposed to the conventional pretest and posttest, which would be given out twice (before and after the intervention). Three important benefits of retrospective pretest measurements are that students may not be available to complete the pretests and posttests, and students tend to overrate their knowledge prior to exposure to the content, known as "response shift bias" (Nielsen 2011). Lastly, this instrument allows occasional learning to be dedicated to delivery of instructions rather than assessing students. This measurement instrument has been useful in keeping record of the changes that occur as a result of students who were self-assessed after exposure to an intervention.

There are two important theories that govern the measurement of competence, which are Classical Test Theory (CTT) and Item Response Theory. The CTT is a theory about test scores that introduces three essential concepts, which are observed score, true score and error score. This theoretical framework has led to the formulation of different models of diverse forms. For example, "the classical test model" is a simple linear model that postulates linking of the observable test score to the sum of two unobservable variables, true score and error score. The basic assumptions in the classical test model are that true scores and error scores are uncorrelated, that the average error score in the population of examinees is zero, and that error scores on parallel tests are uncorrelated. The item response theory is a general statistical theory about examinee item, test competence, and how competence relates to the abilities that are measured by the items in the test. Item responses can be discrete or continuous, dichotomously or polychotomously scored, and can be ordered or unordered. There can be one ability or many abilities underlying test performance, and there are multiple ways in which the relationship between item responses and the underlying abilities can be specified. Ayob *et al.* (2011) mentioned that one-parameter item response theory can quantify the capability of a student based on particular learning results. The item response theory was used to implement a fully adaptive assessment framework for a computer-programming test (Ivančević 2014). This model gave the lecturers enough data to determine students at risk in computer-programming, and it is a useful tool to determine the competence of each student on each question.

2.6 Computer-programming Competence

Computer-programming competence often requires a student to be able to perform a number of programming activities, such as problem and system decomposition, communication, code organisation, code readability, defensive coding, error handling, use of existing frameworks, integrated development environment and application programming of an interface, as well as being able to translate requirements to specification and design databases. An individual who had gained computer-programming competence has the essential abilities to utilise web devices and computerised devices to solve problems (Pérez and Murray 2010). Modern psychology has made a good effort to classify competence into four ladders, which from the lowest level to the highest level are: unconscious incompetence; conscious incompetence; conscious competence; and unconscious competence, respectively.

In the context of computer-programming competence, the "unconscious incompetence" level indicates those individuals who do not understand or know how to do programming and do not necessarily recognise the deficit (Fischer 2011; Cutrer, Sullivan and Fleming 2013). These individuals are not aware that they don't know how to do computer-programming. At this level, students are not aware of their lack of computer-programming skills. They may enter the first day in class with a high level of confidence that far exceeds their abilities, so they may deny the usefulness of the needed skill. These individuals have to recognise their own incompetence and the value of the new skill, before moving on to the next level of competence. The length of time an individual spends at this level of ignorance depends on the strength of the stimulus to learn. The "conscious incompetence" level indicates those individuals who do not understand or know how to do computer-programming, but these individuals recognise the deficit and the value of acquiring new skills in addressing the deficit (Fischer 2011; Cutrer, Sullivan and Fleming 2013). These individuals know or recognise that they don't know, so they find that there are computer-programming skills they need to learn. The students may be surprised to discover that there are so many computer-programming concepts to learn. As they realise that their computer-programming abilities are limited, their confidence drops. During this stage, the students go through a bumpy period, and require plenty of instructions and support from the teachers and institution.

The "conscious competence" indicated individuals who understand or know how to do computer-programming (Fischer 2011; Cutrer, Sullivan and Fleming 2013). These individuals know that they know computer-programming, and are willing to acquire new skills and knowledge. They practice their learning and gain confidence in performing the computer-programming tasks they are involved. They concentrate on the performance of the computer-programming activities given to them. They are aware of their new computer-programming skills, and work harder on refining them. "Unconscious competence" indicates those individuals who have had so much practice with a skill that it has become part of them (Fischer 2011; Cutrer, Sullivan and Fleming 2013). These individuals don't know that they know, where acquiring new computer-programming skills becomes a ritual. The students perform the computer-programming task and can multitask without conscious effort, and with automatic ease. These individuals are able to teach the skill to others, depending upon how and when it was learned. Computer-programming and design skills were extremely important competences that students ought to gain, because they would be able to innovatively think, design and solve real-life problems (Mathews 2010; Mohammed and Mohan 2010). Students who acquired or developed their computer-programming competence, had gained skills and knowledge of computers, programming and steps to solve challenging problems (Jurado, Redondo and Ortega 2014).

2.7 Methods of Improving Computer-programming Competence

The emergence of agile software development technologies such as Scrum, extreme programming, and personal software processes, has shifted the attention from the organisational level of competence improvement to the individual software developer, to take responsibility of competence improvement. The agile software development process places emphasis on self-directed development teams and relies heavily on the abilities of individual software developers to make informed choices about the development methodology they intend to use (Dingsøyr *et al.* 2012). However, agile development methods do not provide the required guidance on how to develop and maintain such a competence (Abrahamsson *et al.* 2002). Extreme programming is a software development technology that includes the client in every stage of the design process (Wood, Michaelides and Thomson 2013). The agile and conventional software engineering technologies were compared and the extreme programming technology was rated the best in

terms of productivity and quality (Wood, Michaelides and Thomson 2013). Extreme programming is a very powerful agile software development method that involves extreme programming practices, which provides support for the engineering techniques (Chen and Wu 2015). Scrum was first used by Ken Schwaber in 1996 (Harvie and Agah 2016). It is an agile method that is supported by good management techniques and is one of the most extensively used methods (Chen and Wu 2015). This method allows for the design of applications in sprint, where each sprint is continuously assessed, and where changes are made before moving to the next sprint (Harvie and Agah 2016). The main focus of the software project is quality, and using the Scrum agile method, where every sprint had an iterative process and continuous feedback, which is proved to be successful in developing the project (Harvie and Agah 2016).

The personal software process is an accepted method for improving software processes at the individual level (Zhong, Madhavji and El Emam 2000; Unterkalmsteiner *et al.* 2012). It can be summed up as the competence of an individual software developer to learn to control and to develop his own development processes (Zhong, Madhavji and El Emam 2000; Unterkalmsteiner *et al.* 2012). It consists of a set of methods, forms and scripts that show software developers how to plan, measure and manage their work effectively. Consequently, it is an accepted method for improving the competencies of an individual software developer. The method is usually taught to students in the form of an educational course with a number of programming assignments (Unterkalmsteiner *et al.* 2012). Students who use this method develop their personal defect and design review checklists, based on their historical defect data. However, the process requires advanced knowledge of process discipline, measurement, planning, estimating, quality management and design that limits its use by first year computer-programming students.

2.7.1 Student-centeredness learning

There is an increasing emphasis on providing higher education that implements a student-centred approach to teaching and learning. This approach has been shown to be the link between transforming students and the teachers (Blackie, Case and Jawitz 2010). In a student-centred classroom, students and instructors actively share the focus. A student-centeredness approach begins with the students and aims to provide an environment where the student can become a mature, fully functioning member of the society through managing learning (Blackie, Case and

Jawitz 2010). Atif (2013) makes reference to an educational psychologist who suggested that learning should be student-centred, arguing that the lecture-based classroom merely reiterated what was in the textbook, and that this medium of teaching needs to be changed. Serving as a motivation to the study reported in this dissertation, using a blended learning based framework, Atif (2013) proposed a conversational model that allows students to communicate with peers in all learning situations.

2.7.2 Peer learning

Pair programming learning is traditionally a common phrase used in the agile software work environments that refers to learning from a peer. It is the practice of two programmers sharing one workstation to concurrently develop software. This approach to problem-solving and software development builds enthusiasm for the problem under investigation, it supports learning from peers, it facilitates effective knowledge sharing and fosters collaboration (Schumm *et al.* 2012). There were many peer-learning techniques used in the past to improve computer-programming competence.

Pair-programming encourages students to work together as a team, programs are solved quicker and more efficiently, morale of students was improved as they are excited and enthusiastic about the future exercises, where there was continuous learning from one another, as well as less errors in computer programs, which improves the quality of these programs (Schumm *et al.* 2012). Pair-programming allows students to work with their class mates to solving a real-life programming problem; peer-learning allows the students to get support from their lecturer and the peers are able to highlight the problems each student is experiencing while trying to solve and understand what the questions are asking of them. The quality of software products and productivity was improved by using pair-programming.

Peer-learning allows the students to gain support from their lecturer, and is able to highlight the problems each student was experiencing, while trying to solve and understand what the question is asking. These authors found “peer instruction” supported a student-centred learning environment, where this new approach has been created since it gave students some relief from the traditional method of teaching (Porter and Simon 2013). These are some of the learner-centric approaches that include collaborative learning, active learning, cooperative learning,

guided inquiry learning, and problem-based learning (Porter and Simon 2013). Peer instruction was implemented in lecture halls, and with relatively fewer course modifications than other techniques (Porter and Simon 2013). There was constructive influence, as peer teaching was used, and there was sufficient proof that comprehension of this knowledge had taken place (Porter and Simon 2013). The peer teaching approach involved learners needing to peruse the course content prior to attending the class and during class lectures, where the learners may collaborate and inquire about the more challenging content that was not understood in order to solve a problem. This class therefore did not use the conventional approach, but rather a group collaborative approach had proven to have positive effects in the computing courses (Porter and Simon 2013). Since learners had been able to pass the computer-programming subject, they did not drop out of the course, and acquired computer-programming competence (Porter and Simon 2013).

2.7.3 Game Learning

"Game-based" prototypes had proven to have a positive effect in the teaching and learning of programming modules (Kuk *et al.* 2014). Kuk *et al.* (2014) had found that modules involving collaboration and engagement had supported students to learn and understand concepts in a subject area. Many researchers found that programming was very complex and boring. Since students thoughts and actions were centred around the technology gadgets and the online resources (Kuk *et al.* 2014). Therefore, students found programming codes very difficult to understand. A "learner" prototype is classified as communication between the student and the facilitation prototype (Kuk *et al.* 2014). Learners were more comfortable and felt that it was simpler to study conceptual knowledge resources so as to better their mark of the last test via a hypermedia software applications, since their comprehension and understanding of the knowledge innovations came by means of self-discovery and group collaboration (Kuk *et al.* 2014). Learners do not want to peruse any resource documentation or attend lectures, as it is too challenging to do so; they want to be actively working and engaging, therefore the module had to be interactive (Kuk *et al.* 2014).

2.7.4 Web Learning

There are many types of learners that enrol in tertiary institutions, however it is practically impossible to be able to foresee the type and level of each individuals skills as soon as they enter a class (MacDonald and Creanor 2010). MacDonald and Creanor (2010) have said that web-based education had been a very adaptable and open environment in which to learn during any period. Many learners prefer to communicate with their friends and family using mobile or web-based community interactive prototypes (MacDonald and Creanor 2010). Learners could collaborate among their clusters, in a web-based environment, which connected students for learning purposes (MacDonald and Creanor 2010). Web learning offer students assistance to collaborate within their clusters which had been planned by the higher education institution to facilitate interaction (MacDonald and Creanor 2010). Many universities have web learning environments that are already created and they allow students to be almost immediately connected to the web learning environments as soon as they register (MacDonald and Creanor 2010). Therefore, deliberations can start taking place to focus on problematic subject areas, challenges that students may encounter, and also to discuss skills among students (MacDonald and Creanor 2010). Certain modules that students study at universities stipulates a requirement that students need to be able to engage within a small group for the completion of certain tasks or activities in that class (MacDonald and Creanor 2010). MacDonald and Creanor (2010) mentioned that learners were using the discussion forums to collaborate on a web learning environment, therefore students would be constantly engaging and re-working their tasks to ensure accuracy. Serrano-Cámara *et al.* (2014) found that the European Higher Education Area identified problem-based learning, project-based learning and case-based learning as different teaching methods that can be used to ensure students engage and collaborate with other students. Higher education institutions attract a diversity of students, and in order to accommodate their different learning needs, they have little choice but to adopt initiatives that provide mechanisms for more flexibility and engagement (Zacharis 2012). Zacharis (2012: 171) notes that "as technological innovations in the form of social media such as blogs, wikis, podcasts, Really Simple Syndication (RSS) feeds and social tagging become increasingly central to the functioning of modern society and to students' daily lives, there is growing interest in their use in formal education applications".

2.7.5 Blended Learning

The blended learning-based technique has been extant for about hundred years, proven to have been successful in many research studies (Carter and Hundhausen 2011). Carter and Hundhausen (2011) had documented many studies, which introduced blended learning into their computer science curriculum, for example, Monash University and the University of Victoria. These universities identified the courses that had proven to be a huge triumph, because students used blended learning techniques such as group creation and collaboration, where motivation levels were high, and peer-learning was incorporated into the curriculum (Carter and Hundhausen 2011). The results showed that students chose to use the blended learning techniques instead of the conventional method of instruction (Carter and Hundhausen 2011).

Similarly, Mathews (2010), a teacher at Wisconsin University, developed a studio learning based curriculum called Neighbourhood Game Design Project, to help students to acquire computer-programming competence. This method allowed students to think, decipher, design and critique the prototype (Mathews 2010). However, students used the blended learning pedagogy to create a simulation of an augmented reality game, where they also had to engage with other participants outside of their university. The teacher met with the students each week to assist and discuss possible challenges encountered with the prototype (Mathews 2010).

These authors identified a solution to address the challenges of students not acquiring problem-solving skills, to create "online peer reviews of programming code" (Agarwal and Hundhausen 2010: 263). They felt that this pedagogical approach helped students to identify source code problems quicker, and it assisted them in peer collaboration. However, many students did not wish to join the online peer review, therefore a 10 percent mark that contributed towards their final grades was awarded to all students, who join the online peer (Agarwal and Hundhausen 2010). In a study by Agarwal and Hundhausen (2010), a mentor was assigned to each student to assist in the learning process using the blended learning methodology. Many universities used a blended learning model and it was successful in those courses, as students had a better understanding of the course content. However, there are many key points a lecturer needs to remember so as to ensure students are able to reflect in a blended learning environment. Carter and Hundhausen note that "classroom assignments should primarily be project-based" (Carter and Hundhausen 2011: 106), where learners were urged to engage with each other to work together, and their works should be assessed.

This research study examined the different peer reviews that allowed students to engage and collaborate. Agarwal and Hundhausen (2010) conducted research in a web-based peer environment called OSBLE (Online Blended learning Based Learning Environment), which had positive effects on student learning, since many skills were developed. This electronic environment allowed students to review and assess programming codes written by others. However, it proved to be problematic, because there was only a small number of students who felt motivated to review codes and provide feedback (Agarwal and Hundhausen 2010). Mathews (2010) used a blended learning-based method that included group research, pairing of fellow students, as well as individual and group field research. However, students did not grasp the knowledge immediately, but slowly started working alone and in groups to find the possible solution. According to Matthews, blended learning pedagogy "presented a learning ecology that differed from their typical school experience, many students initially found it difficult to acclimatise" to the blended learning setting (Mathews 2010: 98). These students attended workshops which motivated them enough to go out and research game simulation, where they used the working games from that workshop and created their own games for their fellow students to work through, critique and test. Blended learning comprises of a wide spectrum of learning methods, techniques, resources and it provides the following intrinsic benefits amongst others (Atef and Medhat 2015; Lin *et al.* 2016):

1. It allows teaching, learning and assessment to continue even when the university is closed;
2. it fosters active learning in students because students can communicate their needs to their teachers and peers;
3. it can mitigate the negative effect of poorly designed online courses with high quality instructor led sessions;
4. it can potentially enhance the quality of learning experience amongst students and teachers;
5. it provides the opportunity to support operating in a global context with greater efficiencies; and
6. it promotes student engagement, inspiration and self-regulated learning with inherent provisioning for student-centeredness and improving academic achievements.

In this study, blended learning was implemented to achieve the three modes of operation (Atef and Medhat 2015). These are: the use of the blackboard learning management system to facilitate

course management and resources to support student learning; the use of the blackboard to enrich the quality of the learning experience of students through interactive learning activities such as face-to-face classroom interaction, collaboration and online assessment; and the use of the blackboard to support self-directed learning with the use of interactive and collaborative learning activities. The use of different learning management systems for blended learning has been examined in the literature. In one of the reviews, the Moodle, Blackboard and Open eClass are three widely used learning management systems for supporting blended learning (Kabassi *et al.* 2016). The literature is in favour of the construction of a learning environment to support the learning experience of students. The MetaTutor is a hypermedia learning environment that was designed to detect, model, trace and foster self-regulated learning about the human body system (Azevedo *et al.* 2008). The Adaptive Learning Model (ALMA) is an environment that supports the process of learning and assessment through texts differing, activities corresponding to different levels of comprehension and individualised guidance based on student specific characteristics (Gasparinatou and Grigoriadou 2015).

2.8 Summary

Mastering computer-programming is a major challenge across the globe, because the failure rates are extremely high in computer-programming courses, and this can be attributed to students not comprehending foundational computer-programming concepts. Hatakka, Andersson and Grönlund have noted, "ICTs are said to be able to improve education in many ways, e.g. improve the delivery of education, improve the learning process, students' writing, support more interaction and reduce the teachers' workload" (Hatakka, Andersson and Grönlund 2013: 94). However, each student needs to use the blended learning pedagogy to understand and develop his or her computer-programming competence.

Since many students had not acquired competence before enrolling for the courses at universities, they feel uneasy and hesitant to comprehend the computer-programming courses (Connolly, Murphy and Moore 2009). Many students had difficulty in understanding computer-programming basics, therefore failure rates were extremely high (Mohammed and Mohan 2010). Students need to achieve the deliverable of developing their computer-programming competence within a few weeks of starting the course. This is impossible to achieve, as they have no basic computer-programming competence by means of which to achieve an acceptable level of

understanding in the first few weeks of the semester, where the core requirement for this course is to develop computer-programming competence (Isong 2014). Isong (2014) found that students entering university for the first time find it a challenge to work with a personal computer, comprehend computer-programming ideas, use the programming atmosphere effectively, and engage in computer-programming lectures. The students may encounter problems while learning computer-programming, since the teaching style may need to change, as these two studies suggest (Isong 2014; Reardon and Tangney 2014).

The authors have identified many different mediums, collaborations and global collaborations that students can use to gain computer-programming competence (Cutler and Borrego 2010; Rosli, Ibrahim Teo and Khairol Azmi 2010). A classroom model that uses the principles of Vygotsky in his lecture hall, namely that "learning and development was a social, collaborative activity" (Atif 2013: 416), and that classroom exercises must be based on real life problems and appropriately presented to reality. Fearon, McLaughlin and Yoke Eng (2012: 115) have argued that "group projects provide a forum for experiential and collaborative learning and by their very nature enable a student-centred focus". Learners that formulate groups engage in peer communication within their groups/facilitator and practically solve a problem within a team, which creates an environment that focuses on student centred-ness (Fearon, McLaughlin and Yoke Eng 2012).

CHAPTER THREE: RESEARCH METHODOLOGY

The purpose of this chapter is to present the research methodology that focuses on achieving the research lucidity objectives defined in Chapter One of this dissertation. These research objectives are summarised here again as follows: firstly, to explore how a blended learning pedagogical technique could be implemented in a web-based teaching and learning environment that could help students in designing, understanding, communicating and collaborating to improve their computer-programming competence; secondly, to implement a blended learning pedagogical technique in a blackboard web-based teaching and learning environment to support the development of computer-programming competence of information technology students; and thirdly, to determine whether a blended learning pedagogical technique implemented in a blackboard web-based teaching and learning environment improves the computer-programming competence of information technology students.

The research methodology of this study is principally descriptive and analytic. The descriptive aspect of the research sought to use a retrospective, pretest method to describe the state of the computer-programming competence of the students who participated in this study. The retrospective pretest is a combination of surveys and class tests. Specifically, a survey instrument enables the researcher to determine the prior competence of students whilst a class test reveals the current computer-programming competence of the students. However, the descriptive research alone is insufficient to provide evidence of the extent to which the blended learning technique has contributed to developing computer-programming competence of the students. To address the limitation of the descriptive research, the analytic research used the item response theory, which was employed to augment the capability of the descriptive research. The item response theory was used to evaluate the responses of the surveys and class tests in order to determine whether the blended learning pedagogy provides evidence of developing computer-programming competence of students.

The research approach of this study, therefore, can be classified as quantitative, because it involves the generation of research data in quantitative form that was further subjected to a rigorous quantitative analysis. In addition, the research approach can be classified as simulation because it involves the construction of a blended learning environment within which relevant data were generated. The participating students were exposed to a number of learning activities

within the blended learning environment, where the researcher was able to provide support to the students on demand. The students were in the first year studying computer-programming modules. This chapter discusses in-depth, the pilot test, ethical issues, the survey conducted, target population, sampling frame, and the process for the collection of data.

3.1 Preliminary Research Tools

3.1.1 Measurement Instrument

A survey was used as the instrument to collect data from the target respondents (Trochim, Donnelly and Arora 2014). Trochim, Donnelly and Arora (2014) had defined two crucial methods to process and deliver surveys to respondents, namely a questionnaire and an interview. A questionnaire was used as the instrument to gather the data about the participants in this study. Close-ended questions allow a participant to select a single response from the available choices, thereby providing objective assessment (Graziano and Raulin 2013). There were objective questions in the questionnaire, where respondents selected answers to each question from the available options. There were two sets of questionnaires that the information technology students had to complete before proceeding further with the course and each questionnaire had two components: a survey and a class test component, to be answered concurrently.

The measurement instrument was distributed to the sample respondents to gain information about their background knowledge in computer-programming and identify challenges that students may have encountered during the problem-solving phase for any computer-programming course content. The computer-programming students had the freedom to complete the questionnaires on blackboard or manually fill out the questionnaire. The surveys were initially uploaded onto the blackboard environment. However, students had not completed the survey, because they were first year students studying computer-programming and they were not familiar with this environment. Many students had not even known what the main components of a computer are, nor how to operate it. Therefore, the researcher had decided to manually print the surveys for respondents to complete.

3.1.2 Ethics

The information that was given by each participant was kept confidential and the researcher's responsibility was to make sure that the participants' privacy is guarded. None of the respondents' information was ever and will ever be distributed to anyone. All data collected from the respondents will be kept within a specific timeframe that was stipulated in the research proposal, as approved by the Research Ethics Committee of the higher degree committee of the university.

3.1.3 Sample Size

The sample selected to participate in this research study was a cluster of individuals studying computer-programming courses for the first time. There are two types of sampling techniques, first being the non-probability and probability sampling. Probability sampling involves random selection of students and non-probability does not (Trochim, Donnelly and Arora 2014). Trochim, Donnelly and Arora (2014) mentioned in their book that nonprobability sampling is executed at the initial stages of any "program [sic], intervention or treatment" (Trochim, Donnelly and Arora 2014). The sampling technique used in the current research was a probability sampling technique, because the developed technique was implemented throughout the semester, and testing occurred both during the semester and at the end of the semester.

The population sampled were first year students studying computer-programming courses. These participants were chosen because the failure rates at DUT were exceptionally high, due to students not having any prior knowledge before beginning the computer-programming course. All first year computer-programming students who were exposed to the intervention discussed earlier were tested on varied object-oriented concepts learnt during the course at different stages during the semester. Seventy-three participants from the Department of Computer Science and Department of Information Technology were selected for this study.

Blackboard (see http://www.blackboard.com) is the chosen learning management system that is used at the institution of the researcher. Consequently, it is used in this study to construct the Web based blended Teaching, Learning and Assessment (WebTLA) environment. These students were exposed to the WebTLA environment for the duration of the computer-

programming course. They were using the blackboard learning management system to facilitate their interaction and engagement for this course. They had to fill out the surveys manually, since many students failed to complete the online surveys. They were tested on varied concepts learnt during the course, and the results had showed an improvement in performance compared to the knowledge that existed before training. However, this study used a unique technique to develop computer-programming competence, which incorporated a blended learning-based pedagogy in the blackboard environment.

3.1.4 Sampling Frame

The researcher had initially chosen all first year students studying computer-programming modules in the information technology and computer systems departments. However, the population size would be too large to obtain data from all respondents, since it would be difficult to retrieve. Therefore the researcher had chosen to use two groups of students. The reason the researcher had chosen only two groups of students who were studying computer-programming from the information technology and computer systems departments is to determine whether the intervention would be possible to implement in a class. These two groups had registered for the computer-programming course and it was assumed that they had no prior knowledge of computers or computer-programming. These students were ideal to test the developed technique and determine whether or not it had a positive effect on student's computer-programming skills. The researcher decided to use 73 computer-programming students that were used in the sampling.

3.1.5 Technology Infrastructure

Technology infrastructure involves all necessary technology resources including hardware, software, networks and internet services, which provide a foundation as well as support for the WebTLA environment. All students studying at DUT had access to free Wi-Fi connection on campus and most of these students had access to open laboratories. They could practice and work within their groups to solve challenging problems. Each of these laboratories was equipped with a desktop machine, computer monitor, mouse, keyboard, internet connection and all the software that computer-programming students use in their courses.

3.2 Blended Learning Environment

This study proposes the construction of a WebTLA environment to develop computer-programming competence amongst information technology students. Blended learning has been defined as a technique of meeting the challenges of tailoring learning and development to the specific needs of an individual by integrating technology innovation offered by web-learning, with interaction provided by the traditional contact learning (Shantakumari and Sajith 2015). There are various intrinsic benefits offered by blended learning, and studies have shown that students of blended learning courses have a positive perception of the learning technique (Ginns and Ellis 2007; Shantakumari and Sajith 2015; Wanner and Palmer 2015).

The WebTLA environment was constructed in this study to support the adaptation of the learning environment with the background knowledge and skills of students and involvement of students in learning assessment through scaffolding. The construction of the environment follows the systematic approach proposed for designing blended learning (Atef and Medhat 2015). Figure 3.1 therefore shows the four essential components of the WebTLA environment to be planning, designing, implementing and reviewing.

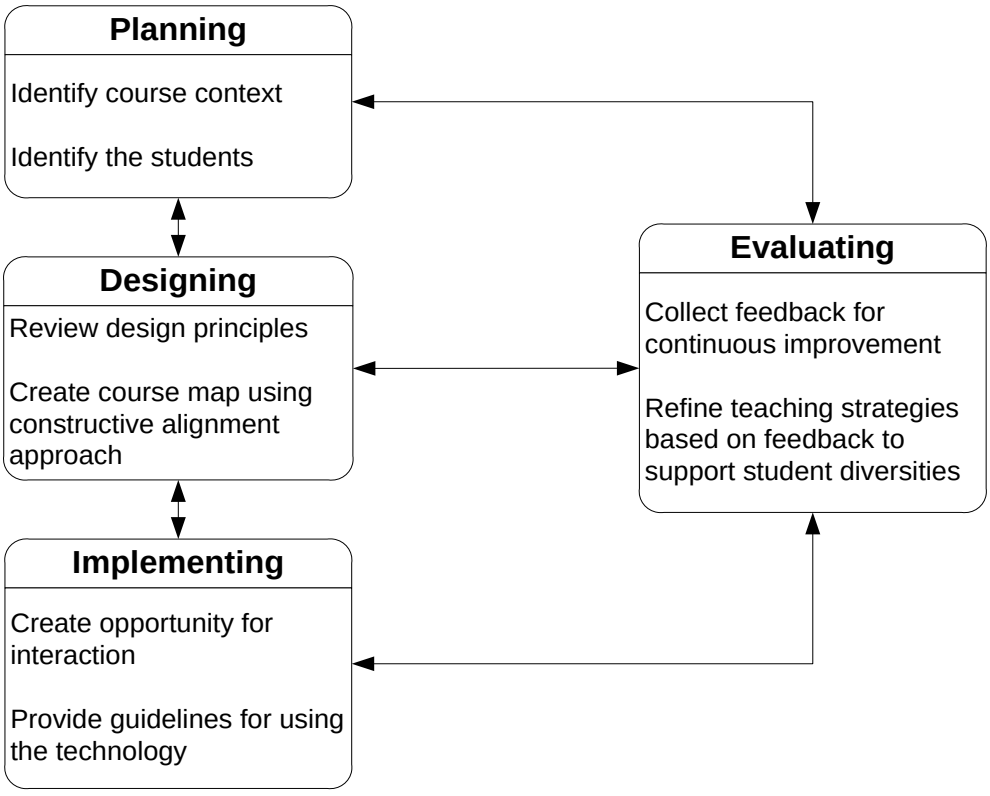

Figure 3.1: Web based blended Teaching and Learning Assessment (WebTLA) environment

3.2.1 Planning

The apparent differential between blended learning and the traditional use of computers in the university system is the anticipated intentional change in the use of technology for education service provisioning. The planning stage of blended learning is crucial to creating conditions for success. In the planning stage, two important decisions must be taken into account, which are identifying the course context by defining course aims and learning objectives, and identifying the students. It is also important to define the relevant knowledge, skills and attitudes or graduate attributes that the course will help the students to achieve. The teaching and learning activities that would best support the learning of students should also be considered in the planning. It becomes easy to find ways of integrating blended learning, once all of the teaching and learning goals and objectives have been planned.

In addition, it is important to identify students by comprehending who they are through their prior knowledge, skills and experiences that they bring to the learning environment. The prior knowledge, skills and experiences of students can help facilitate their assimilation and understanding of new concepts to be learnt (Ranjeeth and Naidoo 2011). Different decision points have to be planned such as deciding on how blended learning can be appropriate for the students, their accessibility to the technology, and the necessary technology training they required to proceed within the environment. Student comprehension identifies the identity of an individual student, including the knowledge and skills of computer-programming and other experiences the student brings to the learning environment that can serve as competence motivation for an individual.

The WebTLA was created because it is extremely important to have adequate background information about a student. The comprehension of the prior computer-programming knowledge and skills of a student ought to have influenced the way in which the teacher plans the teaching of the course. Computer-programming courses, in particular, presents a string of challenges to students, who tend to get bored, and their attentiveness to concentrate is diminished during the class session (Hsieh, Lee and Su 2013). The conventional contact teaching approach, therefore, had proven to be unsuitable to assist students in gaining computer-programming competence. The use of different methods of teaching and learning computer-programming through effective planning may just prove to be successful, as students cannot learn computer-programming in

isolation. They need to engage and collaborate to acquire better understanding of the subject area.

3.2.2 Designing

There is a need to review the design principles used when determining course learning objectives, teaching and learning activities, learning outcomes, and assessment tasks that are required to support student success. The constructive alignment approach should be used to align the course learning objectives, teaching and learning activities, learning outcomes and assessment tasks. All of these should allow students to demonstrate those learning objectives. Literature has provided guidelines to help in the application of the elements of a course design (Atef and Medhat 2015). The following guidelines were applied in the design of the computer-programming course:

a) the WebTLA environment was used for sharing PowerPoint ™ presentations of moderate sizes;
b) the course document was presented in the WebTLA in the form of tutorial guides that were in the format of pdf and accessible online by all students;
c) the active engagement was facilitated in the WebTLA through individual and group collaborative activities;
d) the level of learning that students achieve in WebTLA depends on the assessment tasks; and
e) the collaborative activities of students were facilitated using the discussion forum, online survey and online quizzes that were created in WebTLA.

3.2.3 Implementing

The implementation of the blended learning required the instructor to create an opportunity for the student and instructor to interact. The objective of the implementation stage is to also provide guidelines on how to use the technology. There are various implementation issues that have to be considered, which are infrastructure, integration, professional development and support. All the issues that could inhibit progress or effective functioning such as internet access, power, networking equipment and facilities have to be taken into cognisance.

3.2.4 Evaluating

The evaluation or assessment stage of blended learning is important as a tool for collecting feedback for continuous improvement throughout the course at different points during the course. The model of evaluation can be performed across three areas, which are teaching pedagogies, resources, and delivery strategies (Atef and Medhat 2015). Pedagogies are the learning activities that underpin the course. Resources are the content and information that are provided for the learners, and the delivery strategies are the issues associated with the delivery of courses to students. The assessment of the actual learning, which takes place within the WebTLA, is an important area for this study, which could be regarded as the fourth area of evaluation.

The evaluation of the actual learning that occurs within the WebTLA is generally a significant part of teaching and learning to establish whether or not the learning outcomes were accomplished (Abdulghani *et al.* 2014). Students learnt implicitly, because the instructor had determined the nature and amount of evaluations to assist in refining the current teaching pedagogy. There are a number of evaluation prototypes suggested in courses in academia (Abdulghani *et al.* 2014). The Kirkpatrick model consisted of four stages of evaluation, where every stage of the prototype influenced the following stage (Chrysafiadi and Virvou 2012; Aluko and Shonubi 2014; Dreyer *et al.* 2015). The initial stage concentrated on determining how the respondents handled the course, the following stages assess the learning occurring during the course and the participants satisfaction within the course or not (Chrysafiadi and Virvou 2012; Aluko and Shonubi 2014; Dreyer *et al.* 2015; Fernandez *et al.* 2015).

The evaluation of learning could be conducted before students start the course (pretest) and after taking the course (posttest), which has proven to be unsuccessful to determine the actual learning by students (Dellwo 2010). The unsuccessful result of the pretest and posttest had motivated the researcher to use the retrospective pretest approach. Junker notes “a test cannot be valid unless it is reliable” (Junker 2012: 2). The researcher had decided not to use the conventional pretest and posttest. However, Olugbara *et al.* (2014: 322) study had found challenges with the validity of the measures because of “pretest sensitivity and response shift bias”, therefore a retrospective pretest method was used. Similarly, since there was a problem with response shift bias, where

students overestimate their skills before starting the course. The researcher decided to utilise the retrospective pretest approach to counter the limitations of conventional pretest and posttest.

Nielsen (2011) found that there were two methods, the retrospective pretest and the conventional pretest and posttest. The retrospective pretest design had proved to be successful in assessing people, but the retrospective pretest was given out only once (after completing the course), when comparing it to the conventional assessment, which was given out twice (before and after the course). The retrospective pretest design does not allow participants to overestimate or underestimate their skills, therefore the results produced are reliable. It proved to be successful in linking the skills that a participant had before the intervention and the skills gained after the intervention (Nimon 2013). There was evidence to suggest an increase in skills acquired during the intervention (Nimon 2013). The retrospective pretests were structured so that they cover basic computer-programming concepts and the results compared both the pretest and posttest data. This method determines the type of skills developed to ascertain whether the developed intervention has a positive impact on the computer-programming competence of information technology students.

3.3 Practical Application of WebTLA Environment for Teaching Delivery

The implementation of a WebTLA environment developed in this study was put into practical testing to determine its effectiveness in applying this model, the major steps in planning, designing, in implementation, and in the evaluation stages in the developed model.

3.3.1 Planning Teaching Delivery

The planning stage of the WebTLA incorporated the learning objectives, learning outcomes and the aims for the computer-programming course. The aim of the computer-programming course is to introduce students to computer-programming concepts, problem-solving techniques and practical application of these computer-programming concepts. The computer-programming concepts covered in the course are from object-oriented programming. This approach was used for the benefit of the students, since the instructor referenced objects of the real world and brought the real world objects into the classroom as illustrations. The instructor had also decided to use the object-oriented approach to bring a real world problem for students to solve in the

classroom. The following learning outcomes were identified for the computer-programming course:

a) students must demonstrate an understanding of object-oriented programming concepts;
b) students must apply the concepts of encapsulation and abstraction;
c) students must understand and implement inheritance;
d) students must understand and apply polymorphism;
e) students must read code that can read from and write to secondary storage media; and
f) students must write code that enables communication with hardware.

These outcomes need to be achieved by a student before the end of the semester. Each computer-programming course had learning outcomes for each course area. The following course content was covered for the computer-programming course: Introduction to object-oriented programming, classes and objects, constructors, composition, strings, inheritance and polymorphism. The computer-programming course content referenced many object-oriented concepts and they were extremely difficult for a first year student to grasp, since they had no prior knowledge about computers or basic computer-programming concepts.

There were two class assessments for research purposes that were administered to students. Each of these two class assessments had a pretest survey component. This pretest data was retrieved and used to obtain information about a student's background knowledge on computer-programming concepts. This vital information gave the instructor some guidance about a computer-programming student's prior knowledge. The posttest class test component had tested computer-programming concepts learnt by students during the semester.

3.3.2 Designing Teaching Delivery

Designing is the second stage in the WebTLA model that evaluates the computer-programming course learning outcomes, activities and assessment processes. The learning aims, course objectives and learning outcomes were outlined in the planning stage of the WebTLA. There were two computer-programming courses that the researcher had used in the WebTLA model, the Programming 1 and Programming 2 courses. Students were studying object-oriented

programming concepts, using the learning management system blackboard to facilitate interaction, engaging and collaboration with the instructor. The computer-programming courses were delivered using the C# or C++ computer-programming language/s. The courses had over 80 % of new students, all of whom had not been exposed to any orientation about computers or computer-programming concepts. These students had completed their secondary level education and then enrolled at DUT.

The instructor had the advantage of using student experience and background knowledge about each student to facilitate the use of the four lectures each week. All students engaged in active learning and the instructor focused on special needs of students to improve their performance. The instructor used a traditional pedagogy for formal instruction in the classroom, however a blended pedagogy was used for delivery, learning and assessment. The instructor was well-versed about the subject matter being a computer-programming module. This instructor was capable of stimulating learning among students in a comfortable and stress free environment. Furthermore students had access to blackboard for resources and assistance from their peers and class mates.

All relevant tasks, activities and lessons that the instructor used during lecture time to introduce the computer-programming course content to students is illustrated in Table 3.1 below. The activities and tasks were posted on blackboard, which needed to be completed by students timeously each week. The instructor reviewed the students' answers each week and found that there were some students that struggled to gain the knowledge for that computer-programming concepts. Therefore there was a revision lecture each week to revise previous weeks work in light of students having challenges while trying to solve these computer-programming problems.

Table 3.1 Lessons, activities and tasks performed by students

Serial Number	Lesson	Activity	Task
1	Introduction to OOP	Basic introduction to Object-Oriented Programming concepts in theory to give students an idea of how these OOP concepts relate to the real world.	PowerPoint slides
2	Classes and Objects - PowerPoint slides	Basic introduction to Classes and Objects concepts in theory to give students an idea of how these OOP concepts relate to the	PowerPoint slides

		real world	
3	Creating a Class	A customers' bank account is defined by having a 10-digit customer account number, the customer's full name and an account balance. A customer must be allowed to view the details of their account (account number, name, balance), make a deposit and make a withdrawal. When making a withdrawal, a standard fee of R1.50 per every R100 is applied. This fee must be calculated and deducted from the customers balance.	1. You are required to work as a group using the collaborative tools available in blackboard within your group space. Should you have questions or require assistance from the instructor or other classmates, please post your questions in the General Discussion in the Class Activity thread. 2. Design a solution (header, implementation and driver program) that will create a bank account object and demonstrate the basic behaviour of an account. Include validation of data where appropriate. The concept of encapsulation and data hiding must be demonstrated in your solution.
4	Constructors, Destructors	Introduction to constructors and destructors.	PowerPoint slides
5	Constructors, Destructors	Today's class exercise is a continuation of last week's activity (Account class).	Task1 Read the requirements listed in the **Class Activity - 13 August** link Include a constructor that accepts the account number, name and balance. Task2 Use this new constructor to create an object in the diver programme. Task 3 Include a destructor. Task 4 In the display method, use this pointer to access the properties of the

			object. Task5 In the driver program, create an array of five Account objects. Read details for each customer and display the details of all customers that have a negative balance in their accounts.
6	Composition	Basic introduction to Composition concepts in theory to give students an idea of how these OOP concepts relate to the real world	PowerPoint slides
7	Composition	In December 2014, DUT wishes to honour employees who have worked for the institution. They have different categories of awards/incentives and require a program to assist in determining who receives awards/incentives. The incentives are calculated using the year 2014. **Create a class called Employee that stores the following details:** **Employee Id – must be 5 digits** **Employee name** **Date of Appointment – The date (dd mm yyyy) that the employee started working at the company.** **Type of Employee – Full-time (F) or Contract (C)**	1. Write the default and overloaded constructor for the Employee class. Include set and get methods and a display function that will display the Employee ID and name. 2. Include a function called Duration that returns the number of years that an employee has worked at DUT. This function subtracts the year of appointment from 2014. 3. Also include a function called Convert that will convert a Contract Employee to Full-time if s/he has worked for two or more years (Use the Duration function). The date of appointment must also be changed to 1/12/2012. 4. Write the header and implementation files for the Employee class. 5. In the driver program, create an array of 10 Employee objects. 6. Prompt the user to input details for the employees

			and convert all contract employees who worked for DUT for more than two years to full-time employees. 7. Write a function called **LongService** that accepts the array of employees and displays the details of all employees who have worked for more than 25 years.
8	Composition & Strings	Basic introduction to Composition & Strings concepts in theory to give students an idea of how these OOP concepts relate to the real world.	PowerPoint slides
9	Composition & Strings	Mzanzi Electronics employs part-time workers to assist with maintenance work. All part-time employees start work at 8 o'clock in the morning but can leave work as soon as their task is completed. No employee is allowed to work after 11pm (23h00). The company requires a programme that will store the employees' details and calculate their pay depending on how many hours they have worked.	1. You are required to create a class called Time with properties that store the hour, minute and second in military time. Include the default constructor and the necessary set and get methods. Military time represents time as 00h00 to 23h00. 2. Write code for an overloaded constructor that accepts the hour, minute and seconds as three separate integers. 3. Write a method /member function called NumberofHours that will return the number of hours that a part-time employee worked if all workers start work at 8:00. 4. Create a class called **Employee** (Header, Implementation and Main/Driver). 5. An Employee has the following properties:

			Employee ID (4 digits), **Full Name** (first names separated by a single space. A comma separates the first names from the surname), **Rate** of pay per hour and **End Work** (End work is the time (hh:mm:ss) that and employee finishes work. Overload the Employee class constructor to accept the Employee ID, Full Name, Rate and the time that an employee ends work. This constructor will use the NumberofHours from the Time class to determine the number of hours worked. Include a member function called Display that will print the Employee ID, Surname and initials and the number of hours that the employee worked. 6. In the driver program create an array of 10 part-time employee objects. Prompt the user to enter the Employee Id, Name, Rate and **EndWork** time. Save each employee's details to the array. 7. Display the Employee ID, Full name and pay for only those employees that worked for more than 10 hours.
10	INHERITANCE	Basic introduction to inheritance concepts in theory to give students an idea of how these OOP concepts relate to the real world.	PowerPoint slides
11	INHERITANCE	Create an inheritance hierarchy that a bank might use to	1. Create an inheritance hierarchy containing

		represent customers' bank accounts. All customers at this bank can deposit (i.e. credit) money into their accounts and withdraw (i.e. debit) money from their accounts. More specific types of accounts also exist. Savings accounts, for instance, earn interest on the money they hold. Checking accounts, on the other hand, charge a fee per transaction (i.e. credit or debit).	base class Account and derived classes Savings Account and Checking Account that inherit from class Account. 2. Base class Account should include one data member of type double to represent the account balance. The class should provide a constructor that receives an initial balance and uses it to initialise the data member. The constructor should validate the initial balance to ensure that it is greater than or equal to 0.0. If not, the balance should be set to 0.0 and the constructor should display an error message, indicating that the initial balance was invalid. The class should provide three member functions. Member function credit should add an amount to the current balance. Member function debit should withdraw money from the Account and ensure that the debit amount does not exceed the Account's balance. 3. If it does, the balance should be left unchanged and the function should print the message.

			4. "Debit amount exceeded account balance." Member function get Balance should return the current balance. Derived class Savings Account should inherit the functionality of an account, but also include a data member of type double indicating the interest rate (percentage) assigned to the account. Savings Account's constructor should receive the initial balance, as well as an initial value for the Savings Account's interest rate. 5. Savings Account should provide a public member function calculate Interest that returns a double, indicating the amount of interest earned by an account. 6. Member function calculate Interest should determine this amount by multiplying the interest rate by the account balance. [Note: Savings Account should inherit member functions credit and debit as is without redefining them. 7. Derived class Checking Account

			should inherit from base class Account and include an additional data member of type double that represents the fee charged per transaction. 8. Checking Account's constructor should receive the initial balance, as well as a parameter indicating a fee amount. Class Checking Account should redefine member functions credit and debit so that they subtract the fee from the account balance whenever either transaction is performed successfully. 9. Checking Account's versions of these functions should invoke the base-class Account version to perform the updates to an account balance. 10. Checking Account's debit function should charge a fee only if money is actually withdrawn (i.e., the debit amount does not exceed the account balance). [Hint: Define Account's debit function so that it returns a bool indicating whether money was withdrawn. Then use the return value to determine whether a

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>fee should be charged.
11. After defining the classes in this hierarchy, write a program that creates objects of each class and tests their member functions. Add interest to the SavingsAccount object by first invoking its calculate Interest function, then passing the returned interest amount to the object's credit function.</td>
</tr>
<tr>
<td>12</td>
<td>INHERITANCE</td>
<td>Patient class for the Ethekwini Hospital Accounts Department stores the following

details: a 13-digit ID number, full name, age and amount due to the hospital. Study the header and implementation files for the Patient Class below:

Derive a class called Insured Patient that contains all the details of a patient. In addition to these fields, an Insured Patient must also store the Insurance company ID and the percentage of the hospital bill that the insurance company will pay. Insurance payments are based on the following table:

<table>
<tr><th>Insurer</th><th>Insurer</th><th>Portion of account paid by Insurance (%)</th></tr>
</table>
</td>
<td>1. Include the necessary set and get methods as well as methods to calculate: amount paid by the Insurance company amount due after the Insurance company has paid.
• Include a display method that will print the patients' personal details (ID No., surname and initials and age), the amount paid by the Insurance company and the amount payable by the patient. Write the implementation file for the derived class.
• In the driver program, create an array of 10 Insured Patient objects. Prompt the user to input the details of each patient in the array. Find all those patients who are 50 years and older and are insured by</td>
</tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>a
n
c
e
C
o
m
p
a
n
y

I
D</td><td>a
n
c
e

C
o
m
p
a
n
y</td><td></td></tr>
<tr><td>1</td><td>Old Mutual</td><td>80</td></tr>
<tr><td>2</td><td>Discovery Health</td><td>60</td></tr>
<tr><td>3</td><td>All other companies</td><td>25</td></tr>
</table>
</td><td>Discovery Health.
Display these patient’s details with an appropriate message.
The driver programme must also display the total number of patients insured by each Insurance company with appropriate messages on the screen as shown below:

Total number of patients insured by each company:
Old Mutual = 3
Discovery Health = 5
Other Companies = 2</td></tr>
<tr><td>13</td><td>Polymorphism</td><td>Introduction to polymorphism</td><td>PowerPoint slides</td></tr>
<tr><td></td><td>Polymorphism</td><td>Create a class called fraction that contains a denominator and a numerator data member. These data members must be integers. Write the appropriate set and get functions to set and return the numerator and denominator. Write additional methods to do the following:
Add two fractions and return the result as a fraction.
Subtract two fractions and return the result as a fraction.
Multiply two fractions and return the result as a fraction.
Divide two fractions and return the result as a fraction.
A reduced function that will return the fraction in its simplest form.</td><td>1. Write the statements in the main program that will prompt the user to input the fraction object as two separate integers. Display a menu that offers the user the option to add, subtract, divide and multiply two fractions.
Students also engaged in group discussions online using blackboard. It assisted students within a group to collaborate and solve challenging problems and also discuss their projects. Each student was enrolled into an online group, which was facilitated by the group and they could discuss their projects and problems encountered.</td></tr>
</table>

3.3.3 Implementation of Teaching Delivery

Implementation is the third stage in the WebTLA model that required the instructor to facilitate the active engagement of students and their instructors within the classroom and online. The traditional lectures that were conducted each week by the instructor had been used as follows. Each week the lecturer had four hours of lecture time allocated to the computer-programming class. The lecturer had briefly introduced the main concepts using the PowerPoint presentations available on blackboard during their two-hour lecture time. After the introduction of concepts was completed during this session, students were asked to formulate their own groups, with five students per group. They were given tasks and activities to complete each week.

These class activities and class tasks had to be completed each week in order for students to understand the concept being taught, Table 3.1 shows these activities and tasks. The instructor was present in class during the other two-hour laboratory session to clarify any confusion that students may have had, however there was no formal instruction. They had to use the basics of each section, then do some research to solve the challenging problems within their groups as well as complete the tasks and activities, before moving onto the next section each week. The instructor had reviewed the students' solutions each week to determine if they had understood the concepts for that specific section. After they had reviewed the answers, the instructor used some time that week to revise each course area that students had performed poorly in the activities and tasks.

blackboard is a learning management system that the DUT uses for its online classrooms. Since many students had never used computers or know how to handle a computer, there were guides and videos available for students to use. These were used to understand how to use blackboard and navigate its system. Students were able to review the information that was online as well as to communicate, collaborate and engage within their online classroom. The instructor had created the WebTLA environment using the learning management system blackboard for each computer-programming course. This environment was tailored for the computer-programming students.

It had incorporated resources that were used for sharing among the students. There were PowerPoint presentations, study guides, tutorials, activities, tasks, discussion forums, student project groups were also created. These resources assisted students to understand the instructor's role and responsibility. There were tests that were administered on blackboard for students to complete. These assessments had given the computer-programming students an idea of how to solve a challenging problem. There were group discussions that also took place online, which gave the students freedom to contribute to these discussions at any time. This online component meant that students would not need to be available for face-to-face interaction all the time, which had proven to be time consuming, so the best solution was to use blackboard so that students were able to respond to everyone within their group and everyone did not need to be available at the same time. However, each student has other lectures, assignments to complete and tests to prepare for, so each student responded whenever they had time and they could engage and communicate at any time.

The online discussion forum was used by students to clarify concepts, the comprehension of these concepts and to also give feedback to fellow instructors and students. Each week tasks and activities were uploaded for students to complete. These tasks required students to engage in some research work to find out about certain concepts that the lecturer did not cover during the two-hour session. The lecturer only covered the important concepts of specific course content and they cannot cover everything for that concept. Each task and activity had to be completed by the end of each week so that students could grasp the concept as well as engage in problem-solving and achieve the learning outcome for that week. The lessons and PowerPoint presentation resources were uploaded onto blackboard, which students could then download and use.

The instructor was also available on the learning management system, blackboard, to communicate with students and guide those whom were lost or confused about anything. This meant that they could get assistance from their peer whenever possible. They created discussions within their online groups and their lecturer was enrolled as a peer instructor to lend support wherever needed. This took place in a web-based blackboard environment. All students were encouraged to discuss any challenges encountered when attempting the tutorials and exercises in the teaching and learning environment to benefit all those who are having difficulty with problem-solving. There was a project that covered all sections, which each group had to complete by the end of the semester. This project demonstrated students' level of understanding

and application of knowledge. Students within a group identified problem areas and they were able to resolve these challenges within their group.

For this reason, students had to complete the previous week's work in order for them to understand the following week's work, since it was a continuous process whereby students needed to understand prior knowledge of those concepts to succeed in acquiring computer-programming competence. The instructor had decided to cover the basics of each of the following sections: object oriented programming concepts; objects; classes; methods; abstraction; encapsulation; inheritance; abstract class; polymorphism; functions and data members; constant and volatile functions; pointers and objects; string manipulation; file processing; and interfacing with hardware. Thereafter, students had to work in their groups to discover and collaborate as well as engage in problem-solving to find the solution to these tasks and activities.

The WebTLA was used to facilitate the blended learning pedagogy using the learning management system blackboard. Learning management systems had been implemented in many innovative universities including DUT, where blackboard had assisted students in collaborative and active engagement. This modern technology was used each day for communication between the instructor and students, it was also used to actively engage groups of students using this web-based system. The web-based environment was created for all computer-programming students to collaborate, download and use all the resources that the instructor had made available.

There are graphical user interfaces below with an explanation about the interface. The instructor tried her best to create opportunities for student engagement and collaboration both in the classroom as well as also implementing new technology like blackboard to which the students were exposed, which proved that they were neither bored nor unenthusiastic. They were very interested and enthusiastic about using this technology that enabled them to solve challenging problems. Lastly, they were very excited to communicate using the discussion forums.

Figure 3.2 is a graphical user interface of the training lessons delivered in the blackboard environment. The computer-programming students had access to the blackboard environment at any time and they were interacting and engaging with the environment each day. The instructor was able to access all resources on blackboard, where these students were required to access it continuously to check for any new content or resources that may have been put on the online

environment. The above figure shows all the lessons that the instructor wanted to cover. The first lesson was introduction to Object-oriented Programming, where students were learning about the basics about objects how a computer program is able to be object-oriented. The instructor started with the basics of objects and then the lessons got more advanced based on the student's basic understanding of the initial computer-programming concepts. There were other links on the blackboard environment that students had access to, which were the study guide for the semester, additional tutorials, recommended e-books to use for the course, and several other extra resources that students could use in the computer-programming course.

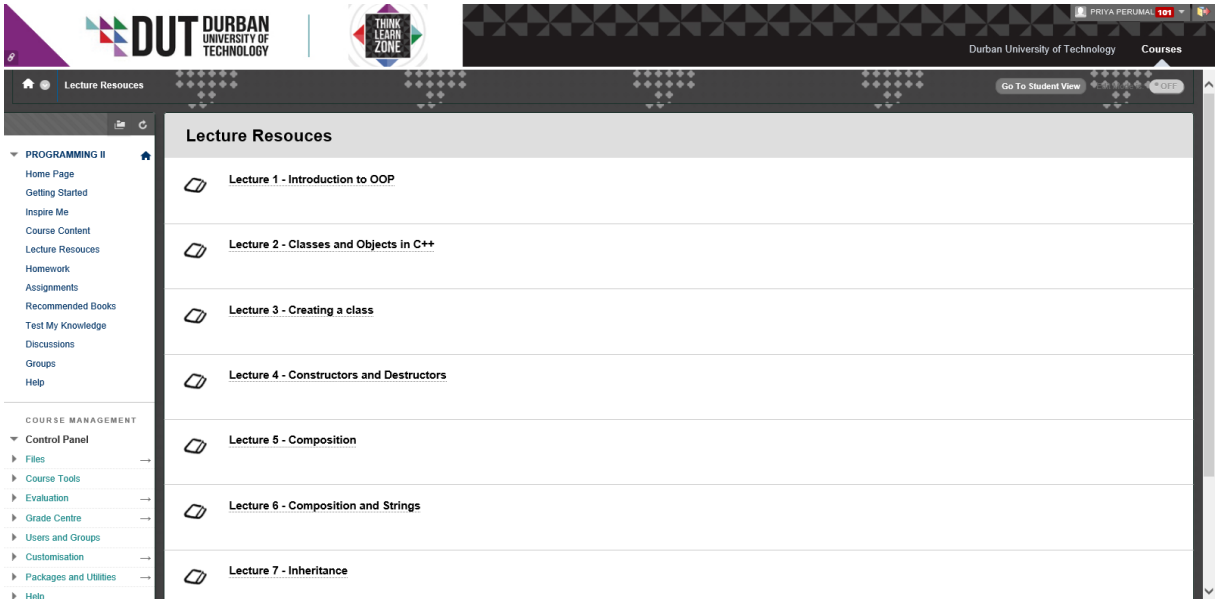

Figure 3.2 Programming 2 graphical user interface of the blackboard environment

In Figure 3.3 the graphical user interface on blackboard was created by the instructor for group interaction. Students had to select their project groups to engage and interact on blackboard. This group discussion was filtered to all students within that group. This gave students a sense of freedom to interact with their group members using blackboard as a medium. The students also did not have to be available for meetings and discussions, since their transition to higher education was extremely stressful. They had a great deal to adapt to at tertiary level and also had to deal with the lectures starting at 08h00 and ending at 16h00. Therefore they could interact whenever they had time within the WebTLA environment.

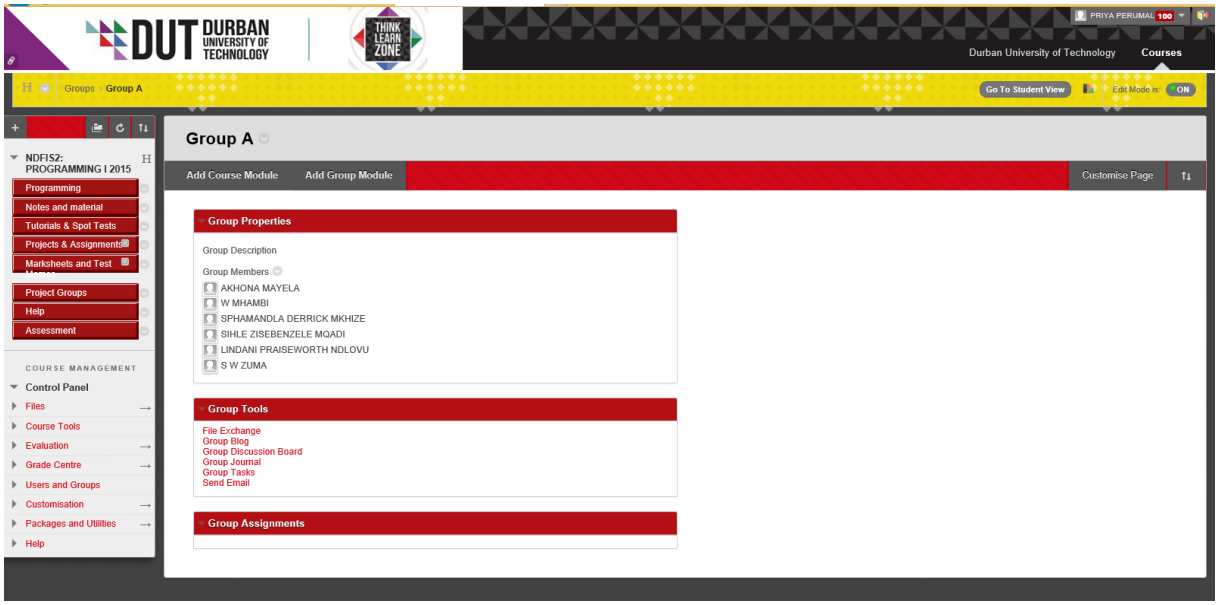

Figure 3.3 Graphical user interface of student groups that were used to create discussion forums on blackboard

Figure 3.4 is a graphical user interface that shows evidence of student's interaction between their group members. Each of the students within the online groups could ask questions, post computer programs online and also comment on any of the other posts that were there. Students had the freedom to talk about the project, how they felt about the tasks being given to them, and their teamwork.

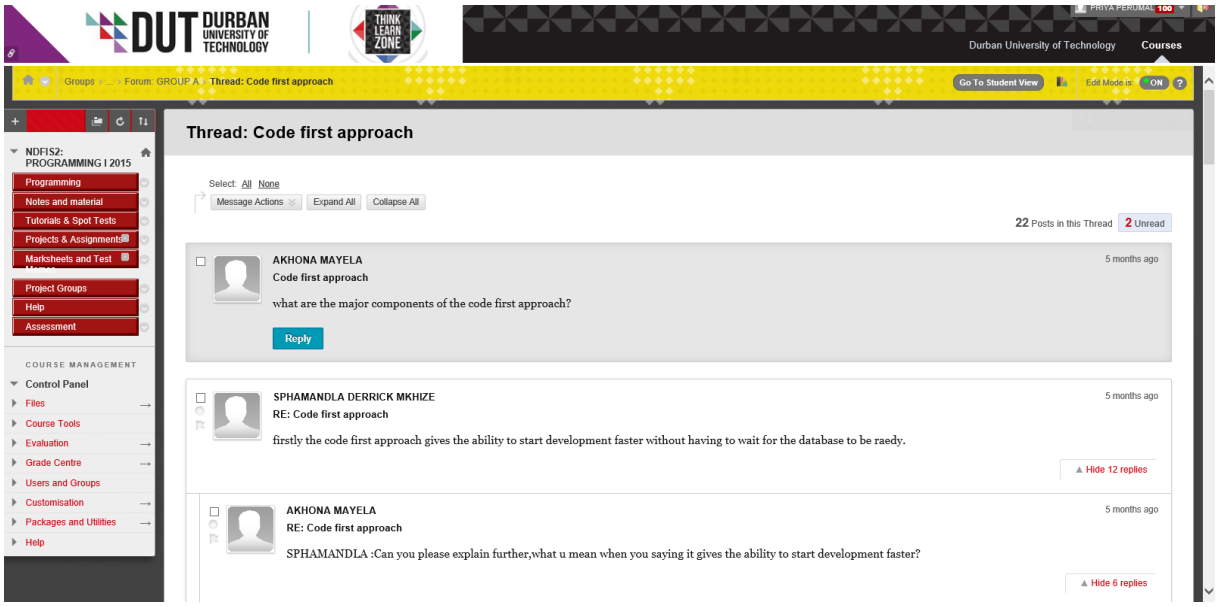

Figure 3.4 Students were using blackboard to facilitate their discussion.

3.3.4 Evaluation of Learning

The evaluation stage involves the collection of feedback from students and refining the teaching strategies based on the feedback received to support students. There were tests at each stage during the semester that students had to complete. Students learnt implicitly because the instructor had determined the nature and amount of class assessments to assist in refining the current teaching pedagogy, where the peer mentorship is a bilateral-approach between teacher and student.

All students were evaluated during each assessment to determine what their strengths and weaknesses were. Thereafter, the lecturer revised problematic computer-programming course concepts after each class assessment, where students had to further complete these remedial exercises for that week before they could continue with the other course content. The instructor decided when and how to alter current teaching pedagogy after the student was assessed. The instructor focused on students' special needs, and sought to engage students in active dynamic learning. After each class assessment, the lecturer had identified the problem areas and had changed the teaching approach to a more student-centric approach, so that students did not feel anxious or stressed. Lastly, teaching pedagogy was refined, identifying challenging course areas after a class assessment, where these students were then required to engage in active dynamic learning.

Mental evaluation instruments have been used to measure the psychological, observable qualities of people and discover different attributes between people (Acar 2015). There are two important psychometric methods that was identified by Petrillo *et al.* (2015) which were CTT and IRT. The Rasch Measurement Theory is a special type of IRT for dichotomous data (Veas *et al.* 2016). The CTT and IRT methods are usually executed on data collected by some means, such as a survey. This research study had looked at the CTT and IRT methods in-depth to determine which of the psychometric methods would be the best to use in this research study. The researcher identified possible problems between the psychometric methods and chose the best possible method that gave individual, in-depth results.

3.3.4.1 Retrospective Pretest

There are two data collection methods identified by Nielsen (2011), first is the retrospective pretest, and the other is the conventional pretest and posttest method. The retrospective pretest was designed in this study to assess the computer-programming competence of information technology students because of its merits as previously discussed. In the design, Questions 1 to 5 constituted the demographic information about each student. Questions 6 to 35 constitute the retrospective pretest, where the first column in the questions signifies the pretest and the questions in the second column signifies the posttest.

The sample questionnaires created using the retrospective pretest method is illustrated in Table 3.2. The retrospective pretest shows pretest and posttest components that basically asks students about computer-programming concepts before starting the course, whether students can solve the problem given for each of the questions. The respondents need to answer either yes, or no. There is no correct answer because the instructor seeks to gain information about the prior knowledge of the computer-programming students. There is also a posttest component on the same assessment that tests the knowledge of students after being exposed to the intervention for this course by requesting the students to objectively solve the problem given and select the correct option from a list of possible answers.

Each posttest has a set of alternatives from which to select, of which one is the correct answer. The reason for the posttest having a correct answer is to determine whether the developed blended learning technique had a positive effect on the computer-programming students. If the computer-programming students got the answer correct to a question that they claimed not to know beforehand, then they would have gained computer-programming competence. The demographic information was included in this survey and class assessment to gain information about the backgrounds of students, where they live, the type of areas in which they live, the language that the students speaks, and whether or not they are living in a university residence.

Table 3.2 Retrospective Pretest questions

1. What is your gender?	☐ Male ☐ Female	
2. What is your age?	☐ 18-25 ☐ 26-33 ☐ 34-39 ☐ 40 or older	
3. Which of the following describes the area you live in?	☐ Urban ☐ Rural	
4. What is your primary language?	☐ English ☐ Afrikaans ☐ Zulu ☐ Other	
5. Are you currently staying at the DUT residence?	☐ Yes ☐ No	
	Before taking the Programming course, I can solve the following problems.	**After taking the Programming course, I can solve the following problems.**
6. ____ are the operations that the object is capable of performing.	☐ Yes ☐ No	☐ Events ☐ Attributes ☐ Behaviors ☐ **Methods**
7. ____ are the actions to which an object can respond.	☐ Yes ☐ No	☐ Attributes ☐ **Events** ☐ Behaviors ☐ Methods
8. Every object has ____, which are the characteristics that describe the object.	☐ Yes ☐ No	☐ Events ☐ Methods ☐ **Attributes** ☐ Behaviors
9. In Object-oriented programming, the problem is divided into _____.	☐ Yes ☐ No	☐ **Classes & objects** ☐ Functions ☐ Structures ☐ Modules
10. A class is _____ datatype.	☐ Yes ☐ No	☐ Primitive ☐ Derived ☐ **User-defined** ☐ None of the above
11. A class is a collection of _____ and ____.	☐ Yes ☐ No	☐ **Data members and member functions** ☐ Data members and main ☐ Data members and include statements ☐ Data members and close statements
12. An object is _____.	☐ Yes ☐ No	☐ **A variable of class type.** ☐ Same as class. ☐ Just like a global variable.' ☐ Collection of data members and member functions.
13. The binding together of manipulated data and functions is known as _____.	☐ Yes ☐ No	☐ Overloading ☐ Polymorphism ☐ Data Abstraction ☐ **Encapsulation**

	Before taking the Programming course, I can solve the following problems.	**After taking the Programming course, I can solve the following problems.**
14. Exposing only the interfaces and hiding the implementation details from the user is known as _____.	☐ Yes ☐ No	☐ Overloading ☐ Polymorphism ☐ **Data Abstraction** ☐ Encapsulation
15. Preventing direct access of data-members of the class from outside interference and misuse is known as _____.	☐ Yes ☐ No	☐ Polymorphism ☐ Encapsulation ☐ **Data Hiding** ☐ Overloading
16. Which header must be included for cin and cout?	☐ Yes ☐ No	☐ stdio ☐ **iostream** ☐ conio ☐ math.h
17. Creating a new class using one or more existing classes is known as _____.	☐ Yes ☐ No	☐ Polymorphism ☐ Encapsulation ☐ Overloading ☐ **Inheritance**
18. When a function can take on different forms, it is known as _____.	☐ Yes ☐ No	☐ **Polymorphism** ☐ Encapsulation ☐ Overloading ☐ Inheritance
19. We can inherit functions and data members from this existing class, which is known as _____.	☐ Yes ☐ No	☐ **Base class** ☐ Derived class ☐ Data class ☐ New class
20. A _____ class can inherit data members and functions from an existing class.	☐ Yes ☐ No	☐ Base ☐ **Derived** ☐ Data ☐ New
21. Which feature in Object-oriented programming allows reusing of code?	☐ Yes ☐ No	☐ Polymorphism ☐ **Inheritance** ☐ Encapsulation ☐ Data Hiding
22. To hide a data member from the program, you must declare the data member in the _____ section of the class.	☐ Yes ☐ No	☐ Hidden ☐ **Private** ☐ Public ☐ Protected
23. the function whose prototype is void getInfo(item * thing); receives ____ to the structure.	☐ Yes ☐ No	☐ **A pointer** ☐ A reference ☐ A copy of the variable ☐ Nothing
24. in the following instruction, Room is a _____. Room objRoom (3,4);	☐ Yes ☐ No	☐ Object ☐ **Class** ☐ Variable ☐ Property

	Before taking the Programming course, I can solve the following problems.	**After taking the Programming course, I can solve the following problems.**
25. in the above instruction, objRoom is the _____.	☐ Yes ☐ No	☐ **Object** ☐ Class ☐ Variable ☐ Property
26. The purpose of a _____ in Object-oriented programming, is to encapsulate the properties that describe an object, the methods that allow the object to perform tasks and the events that allow the object to respond to actions.	☐ Yes ☐ No	☐ Event ☐ **Class** ☐ Method ☐ Attribute
27. The instructions of a ___ are automatically processed each time an object is created from that class.	☐ Yes ☐ No	☐ Structure ☐ Public variables ☐ Properties ☐ **Constructors**
28. A constructor that has no parameters is known as ___.	☐ Yes ☐ No	☐ Parameterized constructor ☐ **Default constructor** ☐ Property Structure ☐ Structure
29. Constructors that contain parameters are known as ___.	☐ Yes ☐ No	☐ **Parameterized constructor** ☐ Default constructor ☐ Property Structure ☐ Structure
30. When two or more methods have the same name but different parameters, the methods are referred to as ___.	☐ Yes ☐ No	☐ Default methods ☐ **Overloaded methods** ☐ Parameterized methods ☐ Property methods
31. Which of the following allows you to create a base class called rectangle and two objects of this class.	☐ Yes ☐ No	☐ Rectangle recA, RecB; ☐ Rectangle ; ☐ **Rectangle recA (3,4); , Rectangle recB (4,5);** ☐ Both A and C

32. Which of the following is correct constructor of the Rectangle class, with the variable *w* and *h*.	☐ Yes
	☐ No
		☐ Rectangle :: Rectangle ( )
		{
		w=A;
		h=B; }
		
		☐ Rectangle :: Rectangle (int , int) {
		w=A;
		h=B; }
		
		☐ Rectangle :: Rectangle (int x , int y) {

		w=A; *h*=B; } ☐ **Rectangle :: Rectangle (int A, int B) {** **_w_=A;** **_h_=B; }**
	Before taking the Programming course, I can solve the following problems.	**After taking the Programming course, I can solve the following problems.**
33. ___ is automatically called when an object is destroyed or the scope of the object has ended.	☐ Yes ☐ No	☐ Constructor ☐ **Destructor** ☐ Parameterized Constructor ☐ Default Constructor
34. Destructors have the same name as the class but are preceded with a ____.	☐ Yes ☐ No	☐ **Tilde sign (~)** ☐ Ampersand sign (&) ☐ Exclamation sign (!) ☐ Hash sign (#)
35. What are the three types of class relationships?	☐ Yes ☐ No	☐ **Generalisation, Aggregation, Association** ☐ Generalisation, Aggregation, Inheritance ☐ Generalisation, Association, Encapsulation ☐ Association, Encapsulation, Data Abstraction

3.3.4.2 Classical Test Theory

One of the traditional methods to assess the validity and reliability of a psychometric test was the Classical Test Theory (CTT) (Cappelleri, Lundy and Hays 2014). CTT displays information that is general and very simple. CTT results display the overall score achieved on a fixed set of elements in a test (Cappelleri, Lundy and Hays 2014; Mallinckrodt, Miles and Recabarren 2015). It assumes that the data is valid, if it was reliable (Junker 2012). However, the estimation of the errors occurs when the true score is made up of the observed score and error score (Acar 2015).

Although CTT was the first method used for psychometric measurement, there were many problems associated with CTT, which can alter the final results (Petrillo *et al.* 2015). These problems were linked to the item-level data linking and the ordered counts rather than the

interval measurement (Petrillo *et al.* 2015). These discoveries produced by CTT were both sample and scale reliant, leading to bad downsides if the estimation performance of an instrument was influenced by the example it should be measuring (Petrillo *et al.* 2015). If facts or statistics were lost, then CTT method cannot ensure that the data was processed properly. Each participant's standard error of measurement scores was presumed to be stable and it did not take into account each participants range on the scale. The Kuder-Richardson scientific equation was used to compute the "Standard Error of Measurement" in CTT (Culligan 2011). However, the Standard Error of Measurement calculates an observed mark that is made up of the true mark and an error mark that is usually disseminated (Culligan 2011).

3.3.4.3 Item Response Theory

Item Response Theory (IRT) is a model based on the possibility of clarifying a participants' response to a specific item (Culligan 2011). IRT is a novel approach, where a participants capability is taken into account and there are three estimation prototypes, "Maximum Likelihood estimator, Bayesian estimator, Maximum a posteriori Probability (MAP)" (Merrouch *et al.* 2014). However the "Maximum Likelihood estimator" was utilised in this study, since the number of mistakes was marginally lower than its rivals.

The Maximum Likelihood estimator is a scientific prototype that had been created to estimate the value of a participants latent trait that looks at the answer a participant has chosen to a specific question and the approach to the IRT, which identified the difficulty of the question, as well as the learner's competence to answer it correctly (Merrouch *et al.* 2014). There was probability of an individual responding to an item based on the skill level (Merrouch *et al.* 2014). This is a good estimation technique used in many disciplines, and it's called IRT. This scientific method had three components to it, the first of which being the prototype estimator, followed by the likelihood of a participant selecting the correct answer, and lastly, measuring a student's capability or skill (Merrouch *et al.* 2014). This approach gained much prominence in education and it used a guessing parameter to determine a student's capability (Merrouch *et al.* 2014).

There are four prototypes of IRT, where the one-parameter IRT Model is often referred to as the Rasch Model because it assumes that the possibility of an odd participant together with the ability to respond to a complex article accurately (Culligan 2011). The equation indicates that the

probability of a correct response is dependent on the ability of the examinee (θ) and item parameter b_i, which is referred to as item difficulty. The higher the item difficulty value is for an item, the more difficult the item/question is and the lower the value for an item is, it means that the item/ question is easier. The probability of a given response is determined based on both the item and examinee characteristics (Foley 2010). The one-parameter logistic model focuses only on item difficulty. The one-parameter logistic equation is:

$$p_i(x_i = 1 | \theta) = \frac{1}{1 + e^{-(\theta - b_i)}} \tag{3.1}$$

The second prototype of IRT is the two-parameter logistic model. This model is an extension to the one-parameter logistic model. It is like the one-parameter model, however, there is an extra parameter, namely, the item discrimination. This item parameter a_i is also known as item discrimination, and it measures the slope of the item characteristic curve at the point of variation (Foley 2010). Item discrimination determines the strength of the relationship between the item response and capability (Foley 2010). This parameter estimates the slope of an item. A higher value would indicate more discriminating items that would translate into better items. The two-parameter logistic equation is:

$$P_i(X_i = 1 | \theta) = \frac{1}{1 + e^{-Da_i(\theta - b_i)}} \tag{3.2}$$

The three-parameter logistic (3PL) model is an extension of the two-parameter logistic model, and allows for the possibility of including a guessing parameter (Foley 2010). The 3PL was used by many researchers who utilised the instrument as part of the evaluation (Foley 2010). Lately IRT models especially the 3PL is gaining widespread acceptance across the globe and among different evaluation prototypes (Foley 2010). These models give an approach to display the likelihood of an examinee giving a right reply to an item (Foley 2010). The 3PL model specifically requires extensive sample sizes to acquire precise parameter gauges (Foley 2010). There are three parameters that this model uses, the first is the b_i parameter, which is the item difficulty and it estimates the difficulty of a question. The second parameter is the a_i parameter, which is referred to as item discrimination or slope. The last parameter c_i is the item lower asymptote or guessing parameter, which indicates the probability of a student guessing the correct answer on an item in the test. This guessing parameter influences the test performance (Tavakol, Rahimi-Madiseh and Dennick 2014). The probability of a correct response in the equation is dependent on the ability of the examinee (θ). The 3PL probability model starts

executing from the guessing parameter, and then is dependent on the other two parameters. This guessing parameter is the probability of a low ability respondent getting an item correct. The three-parameter logistic equation is:

$$P_i(X_i = 1|\theta) = c_i + \frac{1 - c_i}{1 + e^{-Da_i(\theta - b_i)}} \tag{3.3}$$

The four-parameter logistic (4PL) model is a generalisation of the 3PL model, and has an extra parameter that is the upper asymptote for the likelihood of a correct response; it can improve ability estimation, when errors are detected early (Culpepper 2015). There are four parameters for this probability model which are: item difficulty, item discrimination, item lower asymptote and the item upper asymptote (D) (Culpepper 2015; Templin 2016). The first three parameters have been discussed in the 3PL model above. However, the fourth parameter is the D value, which is the item upper asymptote or carelessness parameter (Culpepper 2015; Templin 2016). This fourth parameter makes the model overly complicated; therefore the researcher has decided to reduce it by one parameter and used the 3PL model. The 3PL model is capable of determining the differences between students and their varied computer-programming competences. Therefore, it is a better probability model to use, when we compare it to the classical test theory. The 4PL equation is for the sake of lucidity written as:

$$P(Y = 1|\theta, a, b, c, d) = c + \frac{d - c}{1 + \exp[-1.7a(\theta - b)]} \tag{3.4}$$

3.3.4.4 Measurement of Computer-programming Competence

The measurement of computer-programming competence was done using the survey questionnaire and the class test that was administered to students. Each of the two questionnaires covered two sections, firstly, both the questionnaires had asked for the demographic information about each student. These questions identified what race, age, boarding at university residence or not and first language of students. Thereafter, respondents were asked dichotomous (Yes or No) questions regarding their prior knowledge of each computer-programming concept, which was regarded as the pretest survey in this study. There was no right or wrong answer on the pretest, since this information was required to give the researcher an indication of student's prior knowledge. The second response required from these respondents pertained to the concepts being asked, where students had to answer the objective questions by providing exactly one of

the following answers (A, B, C, D) to the questions. Only one of these responses from the list was the correct answer. There were two different tests given to students and each test had a survey and a class test component, where Table 3.2 shows these components.

Test 1 had covered a few concepts and was given to students after the first six weeks of lectures. This test had a pre- and post- component to it. Each student had to rate their competence before starting the course in the survey questionnaire, and then answer the class test questions. Test 1 questionnaire had about 35 questions, split into five demographic questions and 30 computer-programming concept-related questions. Test 2 practically tested the computer-programming concepts of each of the sections covered during the semester. The Test 2 questionnaire had about 25 questions, split into five demographic questions, and 20 computer-programming questions. These questions had computer-programming codes, and students had to read the code and fill in the blanks for the missing code or answer questions about the computer program. These questions practically tested students on their knowledge about the different concepts of Object-oriented programming practically. Each of these 25 questions had a survey of either a Yes or No answer or an objective class test that had an A, B, C, D answer component that respondents needed to fill in.

The researcher had created questions that were based on the concepts learnt during the semester. The second test was given to students at the end of the semester to test their knowledge of the entire semester, it also had a survey and a class test component. It had tested the students on their application of these concepts, whereby students were given computer programs that they needed to identify certain concepts and fill in the blanks.

The important inspirational idea for the measurement of computer-programming competence is to ascertain that learning has actually taken place amongst information technology students. Traditional class evaluation in a module provides excellent feedback about student satisfaction of learning. However, class evaluations fail to provide an important detail of how much students have learnt. Moreover, class evaluation does not enable students to evaluate their own learning competence. Hence, it is important to find a new way of evaluating students to improve teaching effectiveness. The researcher had taken these limitations of conventional measurement methods into consideration before deciding to create her own approach to determine the change of measurement using the following “ANDNOT” logic gate approach.

This approach measures changes in student behaviour as one of the best indications that learning has taken place and indeed learning is a change in behaviour. The difference between pretest and posttest scores are called gain scores or measurement of change (Dimitrov and Rumrill Jr 2003). These gain scores predict the difference between a pretest score and a posttest score. However, the gain scores have a low reliability (Dimitrov and Rumrill Jr 2003). If a researcher is measuring the change between pretest and posttest scores, then the result being the gain score can be ambiguous, because the difference depends on the items difficulty (Dimitrov and Rumrill Jr 2003). The gain scores do not represent the actual ability on an assessment and it is stated that there is no link between the gain scores and ability scores (Dimitrov and Rumrill Jr 2003). However, the "ANDNOT" logic gate is an attempt to model behaviour change through a mathematical logic. It is based on the following four simple concepts:

a) If a student claims to know an item and if a student gets the answer to the item correct, then learning has not taken place. There is no change in student's behaviour.
b) If a student claims to know an item and if a student gets the answer incorrect, then learning has not taken place. Although there is a change in behaviour. But this is a negative change in behaviour that can lead to dissatisfaction on the part of the student.
c) If a student claims to not know an item and if a student gets the answer incorrect, then learning has not taken place because there is no change in behaviour.
d) If a student claims to not know an item and if a student gets the answer correct, then learning has taken place. In this situation, there is a positive change in behaviour that can create student satisfaction.

The "ANDNOT" Boolean logic operator that outputs 1 only when the pretest input is 0 and the posttest input is 1, otherwise the output is 0 is illustrated in Table 3.3. This type of operator allows the researcher to naturally combine two item response metrics in equal dimension to measure student competence. The reason the researcher had decided to use the ANDNOT approach was to determine the true learning of the students using the retrospective pretest. The pretest results alone would have found out if students had any prior knowledge about the computer-programming concepts prior to starting the course. The posttest would have basically found that the students had gained computer-programming competence. However, using the "ANDNOT" approach will give the results of students who had no prior knowledge, but have

gained some skills after exposure to the blended learning technique. These students had developed computer-programming competence because they acquired correct answers to the items with which they were unfamiliar prior to starting the course, then learning has taken place because of positive change in behaviour. That is what the researcher aimed for this study. To empirically determine whether learning has taken place and in turn develop computer-programming competence of information technology students.

Table 3.3: The "ANDNOT" Learning Decisions for a Retrospective Pretest Examination

Pretest	Posttest	Evidence of Learning	Learning outcomes
1	1	0	No behaviour change
1	0	0	Negative behaviour change
0	0	0	No behaviour change
0	1	1	Positive behaviour change

3.4 Summary

This chapter explained the methodology followed to achieve the study objective. First, it presented the result of the scoping review methodological analysis to determine whether the developed blended learning technique had a positive effect on the computer-programming competence. This chapter described the WebTLA framework that was developed to improve the computer-programming competence of information technology students.

The chapter reported the research methodology approach used in this dissertation. The WebTLA environment that was created had a positive effect on students learning as they had all the computer-programming resources online. They had access to use discussion forums and also collaborate with other fellow students to develop their computer-programming competence. The "ANDNOT" logic gate approach was used to determine whether learning had taken place, this approach also identified students that had no learning taking place, as well as certain instances where a negative behaviour change was identified. This unique technique that was created to ensure information technology students had developed computer-programming competence. The "ANDNOT" logic gate approach is a unique method to determine whether computer-programming competence among information technology students is developed. This approach

was able to combine the pretest and posttest results for each test and display the final reported result. The next chapter presents an analysis of the data and discussion of the results.

CHAPTER FOUR: EMPIRICAL RESULTS

The retrospective pretest data was transferred to the Microsoft excel spreadsheet and uploaded on to the Item and Test Analysis (IATA) software for estimation of IRT parameters. The IATA is a product bundle for the examination of psychometric and instructive appraisal information (Olugbara *et al.* 2014). IRT, which was called a latent trait model that links reactions, for example, to questions, tests or measures and inert attributes, for example, e-aptitudes, capacities, proficiencies or abilities surveyed by a test or scale (Olugbara *et al.* 2014). These factors were input into the Maximum Likelihood algorithm to determine if the technique used in this study had developed students' computer-programming competence. There were pretest, posttest and "ANDNOT" combined test response data for each of the two survey questionnaires. The two groups of computer-programming students' response data were combined into one spreadsheet using the "ANDNOT" logic gate approach.

All demographic data in Table 4.1 included Test 1 and Test 2 results. The analysis of demographic data for Test 1 and Test 2 illustrated in Table 4.1 below, which reflects that a large number of survey respondents' were male (55 percent), when comparing it to the percentage of female (44 percent) students studying computer-programming. Most of the first year information technology students studying computer-programming were between the ages of 18 to 25 with a response ratio that ranges from 92 percent to 96 percent for both Test1 and Test2 shown in Table 4.1. The results reported a 3 to 7 percent other age groups of respondents. Most of the respondents reside in urban areas, with a 60 percent to 65 percent ratio, and the remainder of these respondents lived in the rural areas. Most of the respondent's first language was Zulu and the ratio was 67 percent to 72.7 percent. While other languages accounted for 10 percent to 14.8 percent. Many of these respondents were staying at the DUT residence, which accounted for about more than 50 percent. This is an indication that these students have received funding from the government and, do not have their own computers or laptops to use during their own free time. The results also indicate that these students would need to use university resources like computers, internet, etc. and cannot work whenever or wherever they want to. These information technology students would also only have access to computers and resources on campus, since these resources are not available at the DUT residence.

Table 4.1 Demographic information about information technology students

Characteristic	Category	Test 1 Percentage (%)	Test 2 Percentage (%)
Gender of Respondent	Male	55.9	55,7
	Female	44.1	44,3
Age of Respondent	18-25	92.7	96,7
	26-33	7.3	3,3
	34-39	0	0
	>=40	0	0
Residence Location	Urban	65.5	60,7
	Rural	34.5	39,3
Respondent Language	English	16.4	18
	Afrikaans	0	0
	Zulu	72.7	67,2
	Other	10.9	14,8
Respondent staying at DUT Residence	Yes	61.8	54,1
	No	38.2	45,9

4.1 Pretest Item Result for Test 1

The item statistics identified were discrimination index (Discr), point-biserial (PBis) correlation and item facility (PVal), which were appropriate pointers to determine the usefulness of an item.

a) The discrimination index provides alternative measures of how powerfully connected the responses are to each item, as well as to the actual score of the same relationship. These discrimination index values should be greater than 0.2, therefore the results in Table 4.2 indicate that the discrimination index values are above 0.2. Since many of the discrimination values are high, the item is good at discriminating between high-competence and low-competence students (Olugbara *et al.* 2014).

b) The point-biserial correlation relates to the test scores of an individual on an item and how the responses to each item are strongly interrelated to the other items, as well as the overall test. These values should be greater than 0.2, which would indicate that students with high complete test scores will answer the item correctly, and that students with low complete test scores, will answer that same item incorrectly. All items ranged from lowest of 0.44 to highest of 0.72, which was above the acceptable range illustrated in Table 4.2.

c) The item facility is also referred to as the item difficulty, which describes how easy an individual found an item to be. These item facility values usually range from 0 to 1. However, if the item facility is less than 0.2 or greater than 0.8, then the relationship between

proficiency and performance of the students on a test item is underestimated (Cartwright 2013). An item facility of 0 indicates that no students responded correctly, and a value of 1 indicates that all students responded correctly. The item facility values in Table 4.2 illustrates the values, which are very low, ranging from 0.12 to 0.25, because most students had very little or no prior knowledge at all. The lowest item facility ranges from 0.12 to 0.13. These were documented at Question 14, Question 34, and Question 35.

a. Question 14 - Exposing only the interfaces and hiding the implementation details from the user is known as_____.
b. Question 34 - Destructors have the same name as the class, but are preceded with a _____.
c. Question 35 - What are the three types of class relationships?

Students did not have any knowledge about computer-programming prior to starting the course. The pretest questions only required a yes or a no response. Students may have not understood the actual question and therefore the response was not well documented in the results. Therefore, students' item facility was low, so they did not respond well to these questions. These values indicate that students had no computer-programming competence, therefore they would not have known the concepts covered before. Students were tested on their prior knowledge about computer-programming competence in Test 1 pretest. They did not have any computer-programming competence before commencing the computer-programming course, therefore most of the items on the test were found to be difficult by the students. Their relationship between proficiency and performance on each test item was definitely underestimated.

Table 4.2: IRT parameters that gave the best estimate of computer-programming competence of students in Test 1

Code	Discr	PVal	PBis	a	b	c	Loading
Q6	0.72	0.24	0.77	2.55	0.75	0.00	0.75
Q7	0.72	0.25	0.69	2.01	0.74	0.00	0.67
Q8	0.61	0.24	0.73	2.43	0.76	0.00	0.71
Q9	0.56	0.19	0.73	2.26	0.94	0.00	0.71
Q10	0.61	0.19	0.71	1.88	0.98	0.00	0.69
Q11	0.61	0.18	0.87	4.31	0.92	0.00	0.87
Q12	0.61	0.22	0.68	1.83	0.87	0.00	0.66
Q13	0.50	0.13	0.74	2.42	1.22	0.00	0.72
Q14	0.56	0.15	0.84	3.41	1.09	0.00	0.84
Q15	0.67	0.22	0.78	2.35	0.82	0.00	0.77

Code	Discr	PVal	PBis	a	b	c	Loading
Q16	0.61	0.22	0.72	2.23	0.83	0.00	0.71
Q17	0.56	0.19	0.68	1.58	1.03	0.00	0.67
Q18	0.67	0.19	0.86	3.69	0.87	0.00	0.84
Q19	0.56	0.18	0.84	3.24	0.95	0.00	0.83
Q20	0.61	0.16	0.91	5.26	0.97	0.00	0.90
Q21	0.50	0.15	0.83	3.41	1.09	0.00	0.83
Q22	0.56	0.16	0.82	2.91	1.03	0.00	0.81
Q23	0.33	0.12	0.69	2.24	1.33	0.00	0.68
Q24	0.61	0.19	0.79	2.30	0.94	0.00	0.78
Q25	0.61	0.16	0.85	2.91	1.03	0.00	0.85
Q26	0.56	0.15	0.78	2.62	1.12	0.00	0.77
Q27	0.67	0.19	0.73	1.88	0.98	0.00	0.71
Q28	0.72	0.22	0.78	2.37	0.82	0.00	0.77
Q29	0.56	0.16	0.74	2.33	1.07	0.00	0.73
Q30	0.61	0.18	0.86	3.14	0.95	0.00	0.85
Q31	0.56	0.15	0.80	2.67	1.12	0.00	0.79
Q32	0.61	0.18	0.79	2.09	1.02	0.00	0.78
Q33	0.56	0.16	0.85	2.90	1.03	0.00	0.84
Q34	0.50	0.13	0.80	2.47	1.22	0.00	0.80
Q35	0.44	0.13	0.58	1.23	1.46	0.00	0.57

4.1.1 Pretest Factor Loadings for Test 1

The loadings for Test 1 pretest results were also displayed in Table 4.2 above. These loading range from -1 to 1. The test for validity of the pretest, posttest and combined test response data was performed using reliability and convergent validity estimations. The convergent validity shows the extent to which items represent the same measurement and is measured using the standardised item loadings that should fulfil the 0.4 requirements (Olugbara *et al.* 2014). The results indicated in Table 4.2 shows the correlation between performance on each item and the primary test dimension. The loadings for Test 1 pretest ranges from the lowest, being 0.57, to the highest, being 0.9, which had fulfilled the 0.4 requirement mentioned above. This indicates that the convergence validity is acceptable.

4.1.2 Pretest IRT Parameters for Test 1

The IRT parameters corresponding to the a, b, c parameters in Table 4.2 are discussed below:

a) The a-parameter also referred to as item discrimination, which determines the slope of the Item Response Function (IRF). The larger the value, steeper the slope becomes. This parameter also determines whether an item discriminates between high-achieving students and low-achievers.

b) The b-parameter is also referred to as the item difficulty, which has scores that are scaled as z-scores. A value of three would result in an item being very difficult, a value of zero would result in an item being reasonably difficult, and a value of negative three would result in an item being extremely easy.

c) The c-parameter is a pseudo-guessing parameter that estimates whether a student had no knowledge about the item and would not choose the correct answer.

The IRT a-parameter values for the Test 1 pretest is extremely high, and the slope is a lot steeper. Therefore, this parameter could not discriminate between high and low computer-programming competence students. The IRT b-parameter values in the above Table 4.2 indicate that there were many questions that students found to be difficult. These questions had the highest item difficulty values Q13, Q23, Q34 and Q35 with values ranging from 1.22 to 1.46.

a) Question 13 - The binding together of manipulated data and functions is known as.

b) Question 23 - The function whose prototype is void getInfo(item * thing); receives __ to the structure.

c) Question 34 - Destructors have the same name as the class but are preceded with a ____.

d) Question 35 - What are the three types of class relationships?

All of these questions were very difficult for students to comprehend prior to starting this course. Many of the b-parameter values were over 1, indicating that students found some of the pretest questions difficult to comprehend, since they were not exposed to any knowledge about computers and computer-programming prior to this course. There were no pseudo-guessing as the c-parameter values were all zero.

4.1.3 Pretest Descriptive Statistics for Test 1

The reliability is an indication of the consistency of the test and it also tells us the likelihood of a student repeating the test and obtaining the same score. The reliability of the Test 1 pretest was 0.94 illustrated in Table 4.3, which allowed the researcher to further analyse the response data. There is also a good correlation between the performance on each item and the primary test dimension. The standard deviation is a statistic that displays the scores, based on the mean, where a small standard deviation mean was found to have only a small inconsistency among the students' scores. However, a big standard deviation mean was found to have a huge variability among student's scores, where they would have performed differently on the test. The standard deviation mean was 29.24 for the pretest, indicating that there is minimal variability among student's scores. The response rate was 1.00, which was very good, as all target respondents answered the questionnaire.

Table 4.3: Descriptive Statistics of pretest measures of Test 1

StDev	29.24
Skewness	1.79
Response Rate	1.00
Reliability	0.94
P-75	16.65
P-25	0.00
Median	3.33
Mean	17.89
Kurtosis	1.63
Interquartile Range	16.65
#Respondents	68.00
#OkayItems	30.00
#Items	30.00

4.1.4 Pretest Percent score Chart for Test 1

The percent score chart depicts the number of items a student answered correctly expressed as a percentage of the total number of items administered to the student. The results in Figure 4.1 indicate that students had little or no knowledge about computer-programming before being exposed to the developed blended learning technique, where the percent score graph shows this. The results in Figure 4.1 indicates that a majority of the students had no prior knowledge before

starting the computer-programming course, and that most of the student's scores were between zero and ten percent.

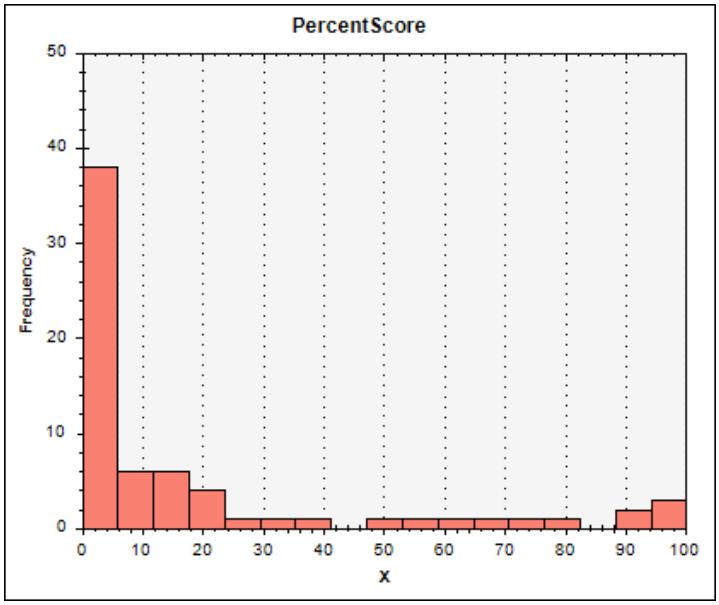

Figure 4.1 – Pretest Percent score Chart for Test 1

4.1.5 Pretest Competence Levels for Test 1

The abilities of all students is shown in Figure 4.2, which illustrates the student's skills were over 1. However, the slope was very steep. Therefore, we could not discriminate between high and low computer-programming competence students. Since many of the students had no prior knowledge before starting the course, this could be the reason that the slope was so steep.

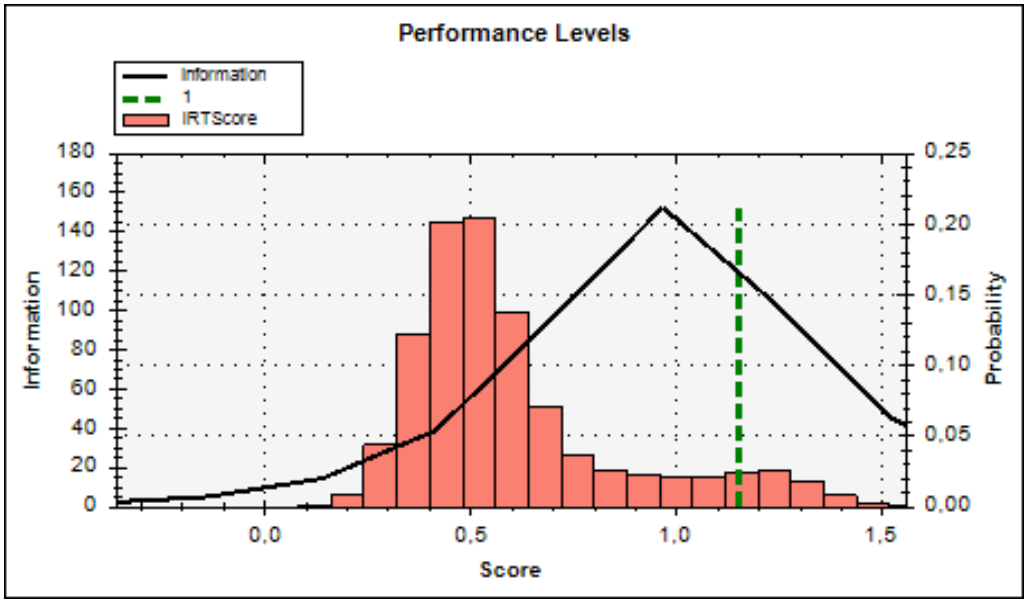

Figure 4.2 – Pretest Competence Levels for Test 1

4.2 Posttest Item Result for Test 1

There were some minor problems in the following questions Q6, Q10, Q11, Q12, Q22, Q26 and Q27. It was noted that the results were not ideal because these questions had minor problems and revisions needed to be carried out for the questions. However, the researcher chose to not revise these questions.

a) Q10 - A class is _____ data type.

b) Q11 - A class is a collection of _____ and ____.

c) Q12 - An object is _____.

d) Q22 – To hide a data member from the program, you must declare the data member in the _____ section of the class.

e) Q26 - The purpose of a _____ in Object-oriented programming is to encapsulate the properties that describe an object, the methods that allow the object to perform tasks, and the events that allow the object to respond to actions.

f) Q27 - The instructions of a ___ are automatically processed each time an object is created from that class.

The questions listed above as needing minor revisions, gives the researcher an idea of students competence in these questions. Students did understand the concepts, however, because of time constraints, it was difficult to learn and grasp the concepts more in-depth. There were small problems that was identified for these questions.

However, the item is not introducing any significant error into the analysis results. The IATA software had found that these questions (items) Q7, Q 13, Q15, Q16, Q23, Q31, Q33, Q34 and Q35 had major problems and needed to be revised. However, the researcher was experiencing problems with the course timeframe, therefore the revisions were made in Test 2. This test comprised of the practical application of the concepts.

a) Q7 - ____ are the actions to which an object can respond.

b) Q13 - The binding together of manipulated data and functions is known as _____.

c) Q15 - Preventing direct access of data-members of the class from outside interference and misuse is known as _____.

d) Q16 - Which header must be included for cin and cout?

e) Q23 - The function whose prototype is void getInfo(item * thing); receives ____ to the structure.

f) Q31 - Which of the following allows you to create a base class called rectangle and two objects of this class.

g) Q33 - ___ is automatically called when an object is destroyed or the scope of the object has ended.

h) Q34 - Destructors have the same name as the class but are preceded with a ____.

i) Q35 - What are the three types of class relationships?

The students may have had problems grasping these concepts, and thus major problems identified by the IATA software. These questions need to be properly revised and discussed with students, as there could be a number of factors that affect these questions. Students may have been confused about the questions because there were four possible answers, and they may have been nervous and forgotten the answer. They could have also not properly understood the question or the possible answers.

The researcher had decided to revise the questions that were problematic in Test 1 and created Test 2, which tested students' practical application of those concepts. Many students had responded well to the questions, when asked in class. Most of the students were tested after taking the computer-programming course. They did not have any computer-programming competence before starting this course, where, as a result, their Test 1 pretest item facility correlations were low. However, they gained valuable knowledge during the course and developed their computer-programming competence. Most of the items on the test were found to be easier than the Test 1 pretest, because students were exposed to the blended teaching and learning technique in a web-based blackboard environment. However, the student's results for Test 1 pretest indicated that most students had no prior knowledge about computer-programming.

These were the item statistics identified as good pointers to determine the usefulness of an item.

a) The discrimination index values for Test 1 posttest shows that 10 questions were below the 0.2 values, which indicates that these questions were not able to discriminate between high and low computer-programming competence of students. These results are shown in Table 4.4. The remainder of the questions were above the 0.2 value.

b) All questions had a point-biserial value ranging from lowest of 0.22 to highest of 0.58 illustrated in Table 4.4. Students responded well to the blended learning technique. They had developed their computer-programming competence, therefore the point-biserial values for most of the items were over 0.2. However there were a few questions that had no values generated for the point-biserial and discrimination index correlation, so these could not determine the relation to the other items and the total score.

c) The item facility values for Test 1 posttest results illustrated in Table 4.4 shows all values that are above 0.2, however, only one item had a score below 0.2. Most students had gained computer-programming competence after being exposed to the blended learning pedagogy. Therefore these values are acceptable and range from 0.22 to 0.72.

Table 4.4: IRT parameters that gave the best estimate of computer-programming competence of students in Test 1

Code	Discr	PVal	PBis	a	b	c	Loadings
Q6	0.29	0.52	0.36	0.44	-0.13	0.00	0.33
Q7	NaN	0.28	NaN	-1.00	-999.00	0.00	NaN
Q8	0.52	0.67	0.39	0.48	-0.96	0.00	0.37
Q9	0.32	0.72	0.27	0.32	-1.91	0.00	0.26
Q10	0.39	0.32	0.36	0.45	1.11	0.00	0.33
Q11	0.29	0.77	0.27	0.35	-2.18	0.00	0.24
Q12	0.12	0.51	0.25	0.21	-0.08	0.00	0.17
Q13	NaN	0.25	NaN	-1.00	-999.00	0.00	NaN
Q14	0.44	0.35	0.37	0.39	1.05	0.00	0.33
Q15	NaN	0.55	NaN	-1.00	-999.00	0.00	NaN
Q16	NaN	0.54	NaN	-1.00	-999.00	0.00	NaN
Q17	0.61	0.58	0.52	1.03	-0.28	0.00	0.55
Q18	0.72	0.39	0.58	1.13	0.36	0.00	0.59
Q19	0.50	0.35	0.54	1.11	0.52	0.00	0.56
Q20	0.61	0.55	0.53	0.99	-0.18	0.00	0.55
Q21	0.55	0.43	0.47	0.80	0.26	0.00	0.50
Q22	0.21	0.52	0.26	0.25	-0.22	0.00	0.20
Q23	NaN	0.22	NaN	-1.00	-999.00	0.00	NaN
Q24	0.42	0.43	0.44	0.57	0.32	0.00	0.43
Q25	0.60	0.46	0.51	0.86	0.14	0.00	0.53
Q26	0.18	0.41	0.22	0.19	1.19	0.00	0.16

Code	Discr	PVal	PBis	a	b	c	Loadings
Q27	0.26	0.23	0.43	0.80	1.17	0.00	0.45
Q28	0.40	0.64	0.40	0.49	-0.77	0.00	0.38
Q29	0.57	0.48	0.55	1.12	0.07	0.00	0.59
Q30	0.51	0.39	0.47	0.64	0.50	0.00	0.45
Q31	NaN	0.32	NaN	-1.00	-999.00	0.00	NaN
Q32	0.35	0.25	0.43	0.61	1.30	0.00	0.41
Q33	NaN	0.13	NaN	-1.00	-999.00	0.00	NaN
Q34	NaN	0.39	NaN	-1.00	-999.00	0.00	NaN
Q35	NaN	0.33	NaN	-1.00	-999.00	0.00	NaN

4.2.1 Posttest Factor Loadings for Test 1

The factor loadings for Test1 posttest displayed in Table 4.4 ranges from the lowest being 0.41 to the highest being 0.59, which fulfilled the 0.4 requirement mentioned above for only the following questions Q17-Q21, Q24-Q25, Q27, Q29, Q30 and Q32. This indicates that there were no correlation between performance and the primary test dimension for the other items that had a loading of NaN (not a number). Since their loadings were below 0.4, the discrimination index and point-biserial values for some questions were calculated and it resulted in a value corresponding to the following NaN, which is a value out of range. Therefore the loadings for these questions were also not calculated, and no correlation between performance and the primary test dimension was determined. There are many reasons why the factor loadings, discrimination index and point-biserial values were NaN, firstly the item results had found these questions to be problematic. This is an indication that the results were skewed and did not give us any actual data to report on. The questions may have been too difficult for students to comprehend, therefore the IATA software found problems with those questions. As a result, the Test 1 posttest results were not good.

4.2.2 Posttest IRT Parameters for Test 1

The IRT a-parameter values for the Test 1 posttest response data was very low and this is an indication that the slope is not so steep. There were no pseudo-guessing as the c-parameter value were zero. However, the discrimination index, point-biserial, item facility and loadings for this test were out of range. This meant that the IRT a, b and c parameter values would not be suitable to talk about, since the other values were not calculated accurately as illustrated in Table 4.5.

4.2.3 Posttest Descriptive Statistics for Test 1

The reliability of Test 1 posttest response data was initially at 0.57. The IATA software generated a list of items that had problems and therefore the reliability was so low. These questions on the Test 1 posttest were excluded from the results Q7, Q13, Q15, Q16, Q23, Q31, Q33, Q34 and Q35. These questions were revised and the researcher included the practical application of these in Test 2. These items could not be included in the analysis due to problems with the data, specifications or the actual question. The researcher decided to remove these items from the analysis as the results generated may be more accurate. Therefore, this resulted in the reliability of Test 1 posttest being generated as 0.71 after removing the possible problem questions. The response rate was 1.00, which was very good as all target respondents answered the questionnaire. The Test 1 posttest data revealed that there should be some suggestion for revision of the test and these questions were reliable.

Table 4.5: Descriptive statistics of posttest measures of Test 1

Mean	47.48
StDev	19.53
Skewness	0.67
Kurtosis	-0.05
Interquartile Range	23.81
P-25	33.33
Median	42.86
P-75	57.14
Response Rate	1.00
Reliability	0.71
#Respondents	69.00
#Items	30.00
#OkayItems	21.00

4.2.4 Posttest Percent score Chart for Test 1

The Test 1 posttest results shown in Figure 4.3 indicate that students had gained some computer-programming knowledge during the first few weeks of the course. The majority of students that are in the x range of 30-65 percent had shown that they were able to develop computer-programming competence.

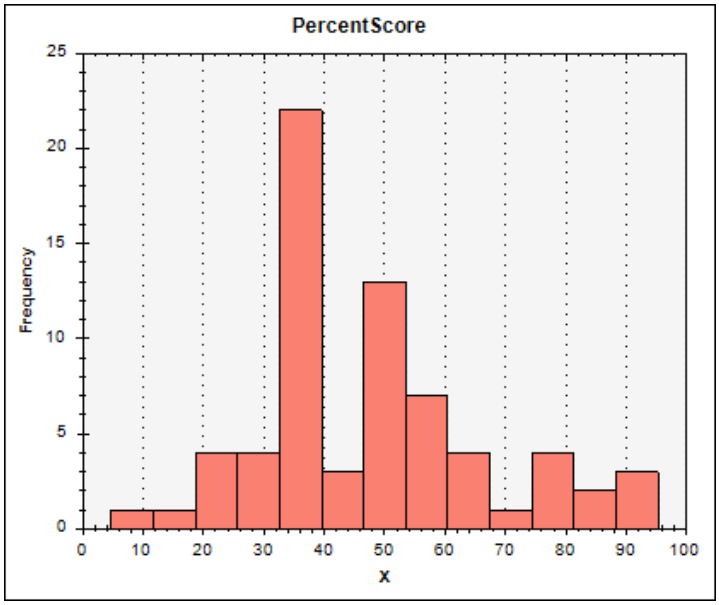

Figure 4.3 –Posttest Percent score Levels for Test 1

4.2.5 Posttest Competence Levels for Test 1

The abilities of all students is shown in Figure 4.4, which illustrates the student's skills were between 0 and 1. This means that the students learning rate had increased and most computer-programming students had gained computer-programming competence. The slope is not so steep, which also indicates that majority of the students skills were between -1 to 1.

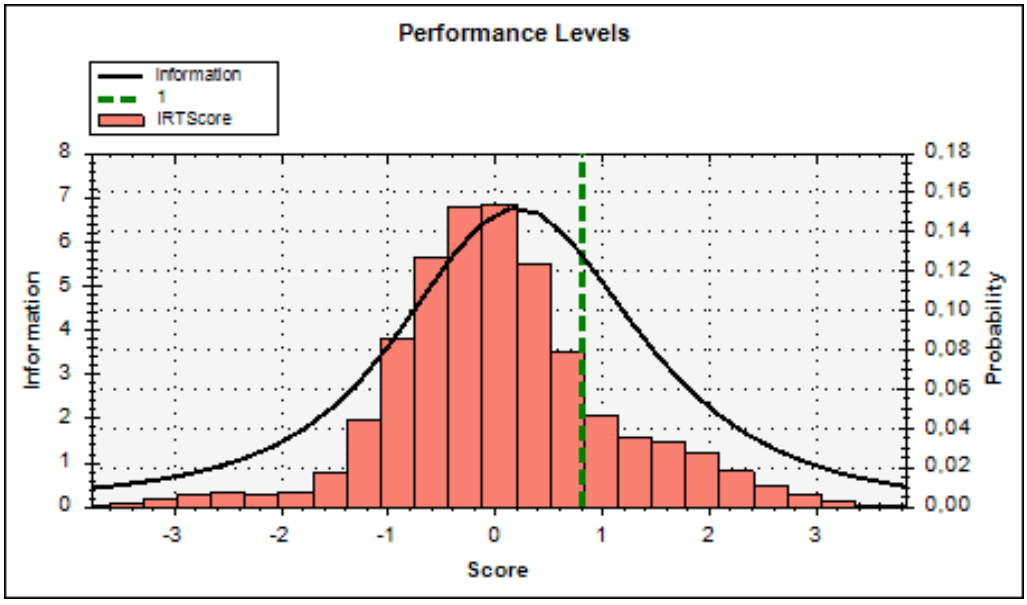

Figure 4.4 –Posttest Competence Levels for Test 1

4.3Pretest Item Statistics for Test 2

The discrimination values for the Test 2 pretest resulted in NaN being replaced for all these questions. However, the point-biserial values for all the items were greater than 0.2. The discrimination index values for Test 2 pretest shows the results in Table 4.6. The item facility results for Test 2 pretest in Table 4.6 shows all items below 0.2. Most students had not gained the computer-programming knowledge before being exposed to the technique. Therefore these values are low-ranging, from 0.12 to 0.19.

Table 4.6: IRT parameters that gave the best estimate of computer-programming competence of students in Test 2

Code	Discr	PVal	PBis	A	b	c	Loading
Q6	NaN	0.23	0.87	3.24	0.74	0.00	0.85
Q7	NaN	0.19	0.91	4.44	0.86	0.00	0.89
Q8	NaN	0.15	0.81	2.45	1.13	0.00	0.79
Q9	NaN	0.16	0.81	3.36	1.01	0.00	0.79
Q10	NaN	0.16	0.86	3.52	1.01	0.00	0.84
Q11	NaN	0.12	0.83	4.07	1.22	0.00	0.82
Q12	NaN	0.16	0.89	3.52	1.01	0.00	0.88
Q13	NaN	0.18	0.93	6.18	0.90	0.00	0.92
Q14	NaN	0.15	0.88	4.76	1.05	0.00	0.88
Q15	NaN	0.15	0.96	8.41	1.02	0.00	0.95
Q16	NaN	0.15	0.82	3.09	1.09	0.00	0.81
Q17	NaN	0.15	0.88	4.76	1.05	0.00	0.87
Q18	NaN	0.12	0.78	5.92	1.19	0.00	0.77
Q19	NaN	0.16	0.88	5.34	0.97	0.00	0.86
Q20	NaN	0.19	0.86	4.44	0.86	0.00	0.85
Q21	NaN	0.18	0.92	6.18	0.90	0.00	0.90
Q22	NaN	0.14	0.92	7.34	1.10	0.00	0.91
Q23	NaN	0.18	0.88	3.91	0.93	0.00	0.87
Q24	NaN	0.15	0.91	8.41	1.02	0.00	0.90
Q25	NaN	0.15	0.88	4.76	1.05	0.00	0.88

4.3.1 Pretest Factor Loadings for Test 2

The loadings for Test 2 pretest was displayed in Table 4.6. The loadings for Test 2 pretest ranges from the lowest being 0.77 to the highest being 0.95, which had fulfilled the 0.4 requirement mentioned above for all the items. This indicates that there was a correlation between performance and the primary test dimension for all items.

4.3.2 Pretest IRT Parameters for Test 2

The a-parameter values for Test 2 pretest response data was very high, ranging from 3.09 to 8.41, and this is an indication that the slope is very steep and the items were able to discriminate between high-achieving students and low achievers. It could also indicate that the questions were very difficult. The b-parameter values illustrated in Table 4.6 indicates that all questions were extremely easy for students, since the values ranged from 0.74 to 1.19. There were no pseudo-guessing as the c-parameter value was zero.

4.3.3 Pretest Descriptive Statistics for Test 2

The reliability of Test 2 pretest response data was initially at 0.93 illustrated in Table 4.7. The response rate was 1.00, which was very good as all target respondents answered the questionnaire. All questions were reliable.

Table 4.7: Descriptive statistics of pretest measures of Test 2

StDev	31.73
Skewness	1.88
Response Rate	1.00
Reliability	0.93
P-75	9.97
P-25	0.00
Median	0.00
Mean	16.01
Kurtosis	1.67
Interquartile Range	9.97
#Respondents	74.00
#OkayItems	0.00
#Items	20.00

4.3.4 Pretest Percent score Chart for Test 2

The Test 2 pretest results indicate that students had no prior knowledge about the computer-programming concepts prior to starting the computer-programming course. Only a few students had a little knowledge about these concepts. The majority of students that fall in the x range of 0 to 10 percent had no prior knowledge of computer-programming competence. These results

illustrated in Figure 4.5 indicate that majority of the students had no prior knowledge before starting the computer-programming course.

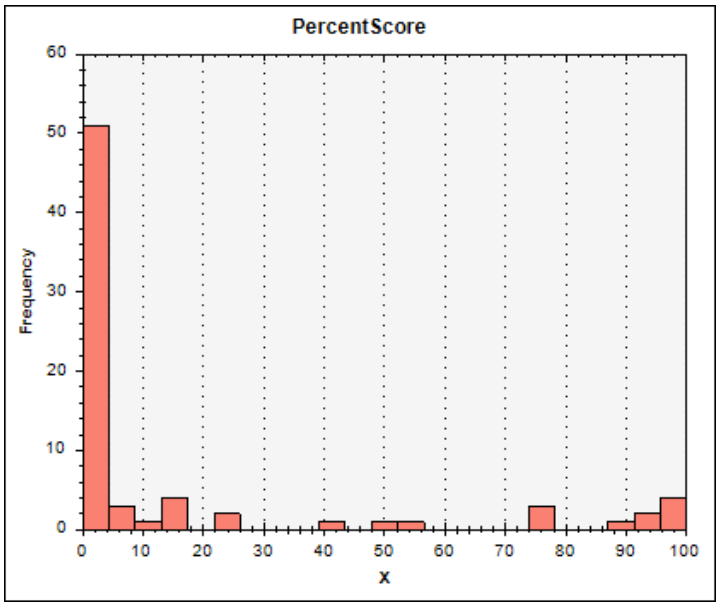

Figure 4.5 –Pretest Percent score Levels for Test 2

4.3.5 Pretest Competence Levels for Test 2

The abilities of all students is shown in Figure 4.6, which illustrates that the student's skills were over 1. However, the slope was very steep. Therefore we could not discriminate between high and low computer-programming competence of students. Since many of the students had no prior knowledge before starting the course, the items were found to be very difficult, which could be the reason that the slope was so steep.

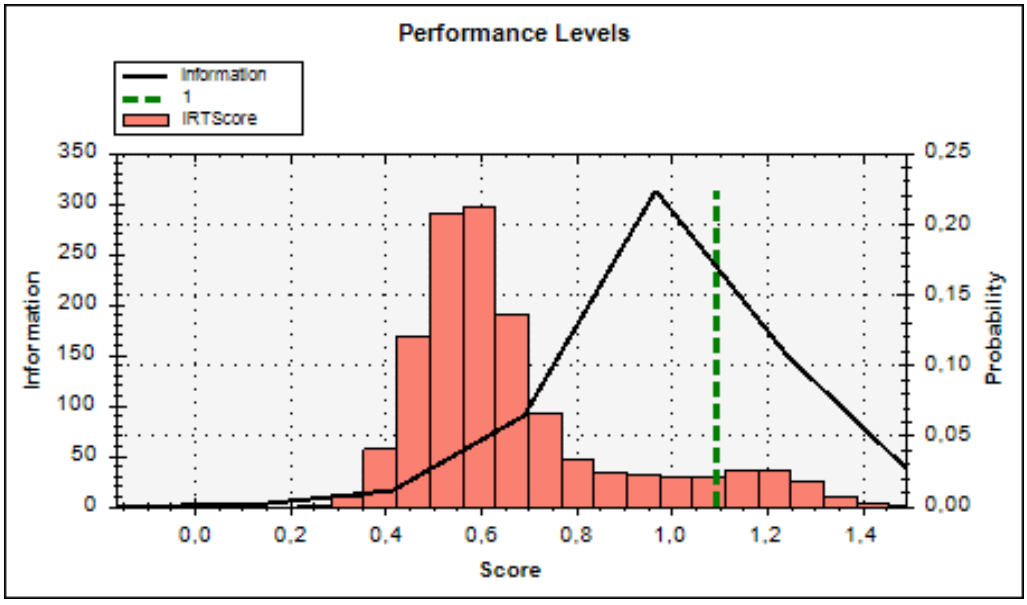

Figure 4.6 – Test2 Pretest Competence Levels for Test 2

4.4 Posttest Item Statistics for Test 2

There were many questions marked as having minor problems, Q7, Q8, Q21, Q23. However, many questions in Test 1 posttest were found to be difficult, which had resulted in values being calculated that were out of range. All the discrimination index and point-biserial values for the Test 2 posttest was above 0.2, as shown in Table 4.8. This indicates that all students' responses were connected to the items. All the discrimination index values were above 0.2, therefore the item is good at discriminating between high-competence and low-competence students. The point-biserial values were able to determine that the students with high complete test scores will answer the items correctly. The item facility values were above 0.2, which indicates that the items were not very difficult.

Table 4.8: IRT parameters that gave the best estimate of computer-programming competence of students in Test 2

Code	Discr	PVal	PBis	a	b	c	Loading
Q6	0.55	0.73	0.43	0.69	-1.07	0.00	0.49
Q7	0.33	0.34	0.32	0.32	1.32	0.00	0.18
Q8	0.31	0.50	0.29	0.25	0.00	0.00	0.23
Q9	0.41	0.24	0.31	0.32	2.25	0.00	0.30
Q10	0.76	0.59	0.62	1.38	-0.29	0.00	0.69
Q11	0.62	0.46	0.52	0.69	0.18	0.00	0.46
Q12	0.55	0.59	0.50	0.95	-0.34	0.00	0.57
Q13	0.64	0.73	0.55	1.24	-0.78	0.00	0.66
Q14	0.61	0.47	0.54	1.09	0.09	0.00	0.61

Code	Discr	PVal	PBis	a	b	c	Loading
Q15	0.71	0.54	0.62	1.37	-0.12	0.00	0.68
Q16	0.58	0.34	0.55	0.76	0.68	0.00	0.45
Q17	0.42	0.55	0.36	0.40	-0.35	0.00	0.37
Q18	0.23	0.39	0.16	0.02	13.86	0.00	0.00
Q19	0.36	0.26	0.24	0.33	2.03	0.00	0.25
Q20	0.56	0.53	0.45	0.44	-0.16	0.00	0.30
Q21	0.30	0.31	0.33	0.54	1.02	0.00	0.31
Q22	0.71	0.45	0.57	1.18	0.18	0.00	0.30
Q23	0.13	0.26	0.24	0.35	1.94	0.00	0.12
Q24	0.53	0.69	0.43	0.56	-0.98	0.00	0.41
Q25	0.43	0.30	0.41	0.58	1.04	0.00	0.31

4.4.1 Posttest Factor Loadings for Test 2

Some of the factor loadings were in the acceptable range of 0.4 and above, however Q7, Q8, Q9, Q17, Q18, Q19, Q20, Q21, Q23 and Q25 were lower than 0.4. These questions had no correlation between the performance and primary test dimension for the above-mentioned questions.

4.4.2 Posttest IRT Parameters for Test 2

Most of the a-parameter values were ranging from 0 to just over 1, indicating that the slope was not very steep, and therefore, that this parameter was able to discriminate between high and low computer-programming competence of students. These results are illustrated in Table 4.8. There were variations of b-parameter values ranging from -1.07 to 13.86. Some questions were reasonably difficult and others were very difficult, or easy. There were no pseudo-guessing as the c-parameter values was zero. The results are similar to Test 1 posttest.

4.4.3 Posttest Descriptive Statistics for Test 2

The reliability of Test 2 posttest response data was initially at 0.72 shown in Table 4.9. Test 1 posttest reliability was 0.71, however there were problems with the results reported. The response rate was 1.00, which was very good, as all target respondents answered the questionnaire.

Table 4.9: Descriptive statistics of posttest measures of Test 2

Mean	46.35
StDev	20.01
Skewness	0.19
Kurtosis	-0.39
Interquartile Range	30.00
P-25	30.00
Median	45.00
P-75	60.00
Response Rate	1.00
Reliability	0.72
#Respondents	74.00
#Items	20.00
#OkayItems	20.00

4.4.4 Posttest Percent score Chart for Test 2

The Test2 pretest results indicate that students had no prior knowledge of computer-programming concepts prior to starting the computer-programming course. However, in Test 2, the posttest results indicate that there was improvement in students' skills. The majority of students' knowledge had increased over the semester, and most of these students had gained computer-programming competence.

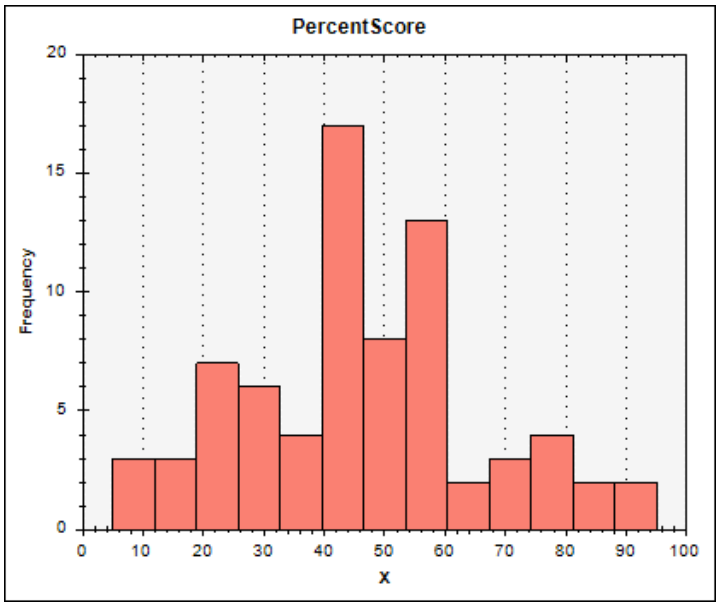

Figure 4.7 –Posttest Percent score Levels for Test 2

4.4.5 Posttest Competence Levels for Test 2

The abilities of all students is shown in Figure 4.8, which illustrates the student's skills were between 0 and 1. This means that the students' learning rate had increased and that most computer-programming students had gained computer-programming competence. The slope is not so steep, which also indicates that majority of the students skills were between -1 to 1, which is a good indication that they had gained computer-programming skills.

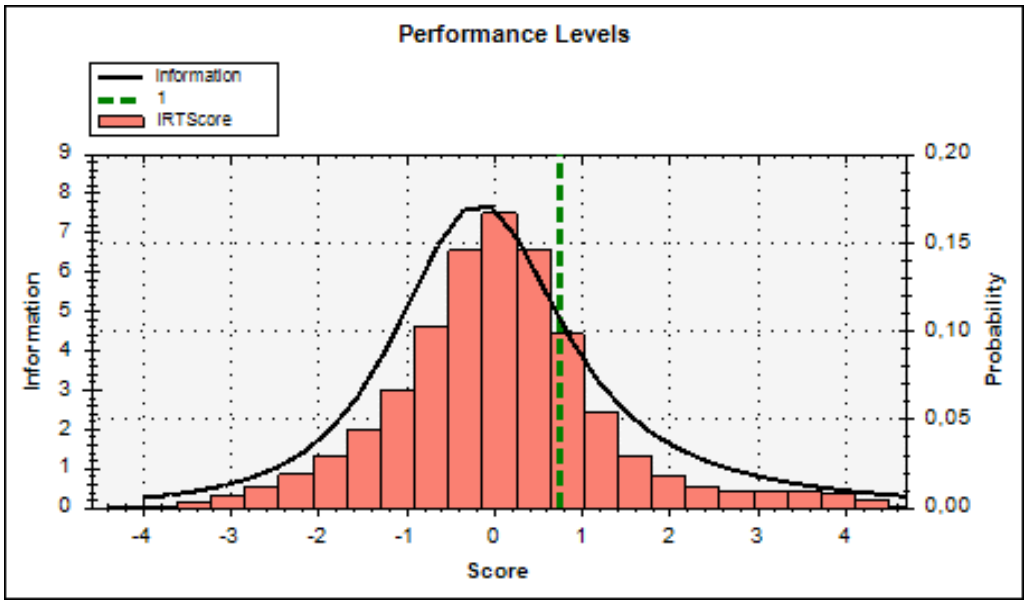

Figure 4.8 –Posttest Competence Levels for Test 2

4.5 Combined Pretest and Posttest for Competence Evaluation

In this dissertation, the researcher had decided to take the pretest results, which were a yes/no (1, 0) response and combine this results with the posttest results. The "ANDNOT" logic gate approach enables the students to evaluate their own learning competence in the competence evaluation model.

4.5.1 Combined Pretest and Posttest Item Statistics for Test 1

There were a few questions marked with a red triangle, which are Q7, Q13, Q16 and Q23. Most of the discrimination index and point-biserial values for the Test 1 combined pretest and posttest was above 0.2. There were only two items that had discrimination index or point-biserial values

less than 0.2. This indicates that most students' responses were well-connected to the items. Most of the discrimination index values were above 0.2, therefore the item is good at discriminating between high-competence and low-competence students. The point-biserial values were able to determine that the students with high complete test scores will answer the items correctly. The item facility values were above 0.2, which indicates that the items were not very difficult. Only one item facility value was slightly below 0.2, namely Q23, which had a item facility of 0.19. This item could have been a little difficult for students, however, it is only 0.1 less than the acceptable range. There were no items that had generated a NaN value, which is an indication that the test was good. The Test 1 and Test 2 results had generated an out of range value for either of the following discrimination index, point-biserial, item facility or loading. Therefore using "ANDNOT" logic gate approach and combining the results of Test 1 pretest and posttest together was successful in developing computer-programming competence. The same was done for Test2, pretest and posttest, which reported good results. The Test 1 combined final results had no out of range values generated and the Test 2 combined final results had no out of range values generated, which made the test results successful in determining whether students had developed computer-programming competence.

Table 4.10: IRT parameters that gave the best estimate of computer-programming competence of students in Test 1

Code	Discr	PVal	PBis	a	b	c	Loading
Q6	0.41	0.38	0.42	0.58	0.59	0.00	0.41
Q7	0.26	0.24	0.22	0.10	6.67	0.00	0.06
Q8	0.51	0.46	0.43	0.58	0.22	0.00	0.42
Q9	0.56	0.59	0.50	0.70	-0.38	0.00	0.44
Q10	0.47	0.24	0.42	0.75	1.20	0.00	0.46
Q11	0.62	0.63	0.55	0.89	-0.50	0.00	0.51
Q12	0.46	0.40	0.39	0.47	0.60	0.00	0.34
Q13	0.10	0.22	0.16	0.10	7.54	0.00	0.07
Q14	0.53	0.28	0.48	0.62	1.10	0.00	0.41
Q15	0.52	0.41	0.38	0.35	0.65	0.00	0.26
Q16	0.15	0.46	0.09	-0.03	-3.33	0.00	-0.10
Q17	0.78	0.47	0.60	1.49	0.09	0.00	0.69
Q18	0.52	0.29	0.53	1.11	0.72	0.00	0.60
Q19	0.52	0.26	0.49	1.12	0.84	0.00	0.57
Q20	0.62	0.44	0.52	0.90	0.22	0.00	0.55
Q21	0.47	0.38	0.43	0.70	0.51	0.00	0.47
Q22	0.41	0.46	0.34	0.50	0.24	0.00	0.35
Q23	0.21	0.19	0.26	0.29	3.03	0.00	0.18
Q24	0.73	0.38	0.58	1.14	0.39	0.00	0.60
Q25	0.79	0.37	0.57	1.02	0.47	0.00	0.59

4.5.1.1 Combined Pretest and Posttest Factor Loadings for Test 1

Some of the factor loadings were in the acceptable range of 0.4 and above, however there were a few that were lower than 0.4. These questions had no correlation between the performance and primary test dimension.

4.5.1.2 Combined Pretest and Posttest IRT Parameters for Test 1

Most of the a-parameter values ranged from 0 to just over 1, and this indicates that the slope was not very steep, and therefore, this parameter was able to discriminate between high-competence and low-competence students. There were variations of b-parameter values ranging from -3.33 to 7.54. Some questions were reasonably difficult and others were very difficult or easy. There were no pseudo-guessing as the c-parameter values was zero. This could be as a result of the "ANDNOT" logic gate approach that was used to combined Test 1 pretest and posttest data.

4.5.1.3 Combined Pretest and Posttest Descriptive Statistics for Test 1

The reliability of Test 1 had dropped significantly from 0.94 to 0.71. This high drop is due to the fact that the Test 1 pretest results showed only the responses of student's knowledge prior to starting the computer-programming course. The Test 1 combined pretest and posttest data is a true reflection of a student's computer-programming knowledge because the "ANDNOT" notation was implemented to determine the learning rate. Therefore the reliability had dropped, since the current combined test measured the learning outcome of each student, using the combination of pretest and posttest data. The response rate was 1.00, which was very good, as all target respondents answered the questionnaire. This result for Test 1 is a true reflection of the student's computer-programming competence.

Table 4.11: Descriptive statistics of pretest and posttest measures of Test 1

Mean	37.72
StDev	19.65
Skewness	0.02
Kurtosis	-0.48
Interquartile Range	25.00
P-25	25.00
Median	35.00
P-75	50.00
Response Rate	1.00
Reliability	0.71
#Respondents	68.00
#Items	20.00
#OkayItems	19.00

4.5.1.4 Combined Pretest and Posttest Percent score Chart for Test 1

The Test1 pretest and posttest percent score results indicate that students had not gained any knowledge during the first few weeks of the computer-programming course. Therefore the x score has a small frequency of students still at 0 percent. However, a majority of these information technology students had gained valuable knowledge as a result of using the "ANDNOT" approach. There was an improvement in students' skills. The majority of students' knowledge had increased over the semester and most of these students had gained computer-programming competence.

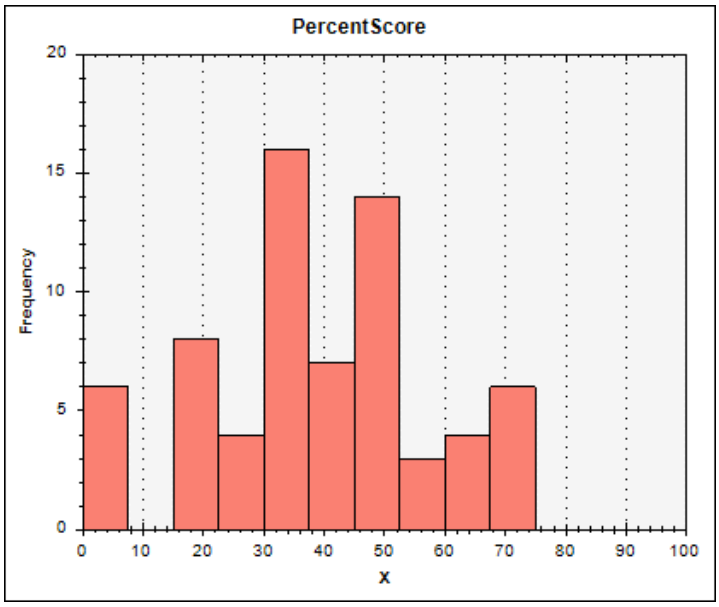

Figure 4.9 –Pretest and Posttest Percent score Levels for Test 1

4.5.1.5 Combined Pretest and Posttest Competence Levels for Test 1

The abilities of all students is shown in Figure 4.10, which illustrates the student's skills were between 0 and 2. This means that the students learning rate had increased and most computer-programming students had gained computer-programming competence. The slope is not so steep, which also indicates that majority of the students skills were good, and an indication that they had gained computer-programming skills.

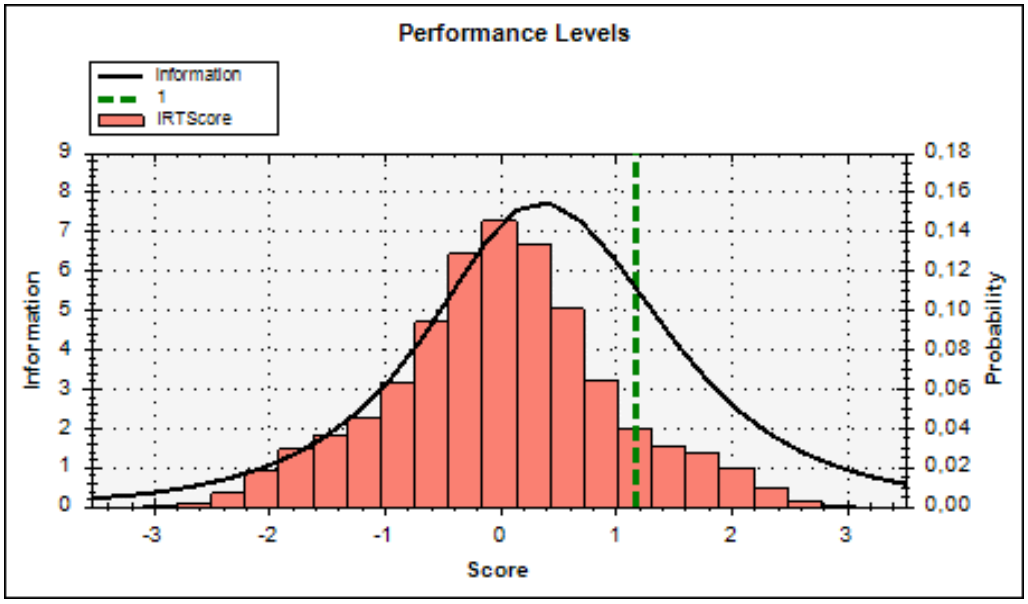

Figure 4.10 – Pretest and Posttest Performance Levels for Test 1

4.5.2 Combined Pretest and Posttest Item Statistics for Test 2

Only one problem was indicated on Q19 as this question was flagged with a red triangle. All of the discrimination index and point-biserial values for the Test 2 combined pretest and posttest were above 0.2. This indicates that all students' responses were connected to the items and the overall score. All of the discrimination index values were above 0.2, therefore the item was good at discriminating between high-competence and low-competence students. The point-biserial values were able to determine that the students with high complete test scores will answer the items correctly. All students answered certain items correctly based on the competence. The item facility values were above 0.2, which indicates that the items were not very difficult.

Table 4.12: IRT parameters that gave the best estimate of computer-programming competence of students in Test 2

Code	Discr	PVal	PBis	a	b	c	Loading
Q6	0.73	0.55	0.59	1.02	-0.19	0.00	0.59
Q7	0.54	0.28	0.51	0.95	0.83	0.00	0.51
Q8	0.56	0.42	0.50	0.74	0.34	0.00	0.48
Q9	0.42	0.30	0.39	0.53	1.11	0.00	0.36
Q10	0.74	0.49	0.62	1.15	0.04	0.00	0.62
Q11	0.49	0.50	0.46	0.63	0.00	0.00	0.44
Q12	0.71	0.45	0.57	0.97	0.19	0.00	0.56
Q13	0.82	0.61	0.68	1.62	-0.31	0.00	0.70
Q14	0.38	0.32	0.40	0.54	0.94	0.00	0.38
Q15	0.75	0.42	0.60	1.14	0.27	0.00	0.61
Q16	0.49	0.28	0.53	0.91	0.85	0.00	0.51
Q17	0.79	0.38	0.65	1.32	0.38	0.00	0.66
Q18	0.44	0.32	0.34	0.41	1.16	0.00	0.32
Q19	0.06	0.12	0.19	0.31	3.96	0.00	0.14
Q20	0.88	0.41	0.68	1.44	0.28	0.00	0.68
Q21	0.45	0.23	0.43	0.69	1.30	0.00	0.40
Q22	0.66	0.50	0.60	1.14	0.00	0.00	0.61
Q23	0.71	0.34	0.53	0.86	0.64	0.00	0.50
Q24	0.85	0.62	0.73	1.99	-0.33	0.00	0.74
Q25	0.83	0.45	0.65	1.26	0.17	0.00	0.64

4.5.2.1 Combined Pretest and Posttest Factor Loadings for Test 2

All of the factor loadings were in the acceptable range of 0.4 and above. This indicates that there was a strong correlation between the performance and primary test dimension.

4.5.2.2 Combined Pretest and Posttest IRT Parameters for Test 2

Most of the a-parameter values were ranging from 0 to 1.99, indicating that the slope was not very steep, and therefore, this parameter was able to discriminate between high-competence and low-competence students. There were variations of b-parameter values ranging from -0.31 to 1.16. Students felt that all questions were reasonably easy. There were no pseudo-guessing as the c-parameter values was zero.

4.5.2.3 Combined Pretest and Posttest Descriptive Statistics for Test 2

The reliability of Test 2 combined pretest and posttest response data was calculated at 0.82. The response rate was 1.00, which was very good, as all target respondents answered the questionnaire.

Table 4.13: Descriptive statistics of Test 2 Pretest and Posttest measures of Test 2

Mean	39.93
StDev	25.37
Skewness	-0.03
Kurtosis	-0.71
Interquartile Range	35.00
P-25	20.00
Median	40.00
P-75	55.00
Response Rate	1.00
Reliability	0.82
#Respondents	74.00
#Items	20.00
#OkayItems	19.00

4.5.2.4 Test 2 combined Pretest and Posttest Percent score chart

The Test 2 pretest and posttest percent score results indicate that students had gained sufficient knowledge during the computer-programming course. Therefore, the x score has a small frequency of students still at 0. However, a majority of these information technology students had gained valuable knowledge during the computer-programming course. There was an improvement in students' skills. The majority of students' knowledge had increased over the semester, and most of these students had gained computer-programming competence.

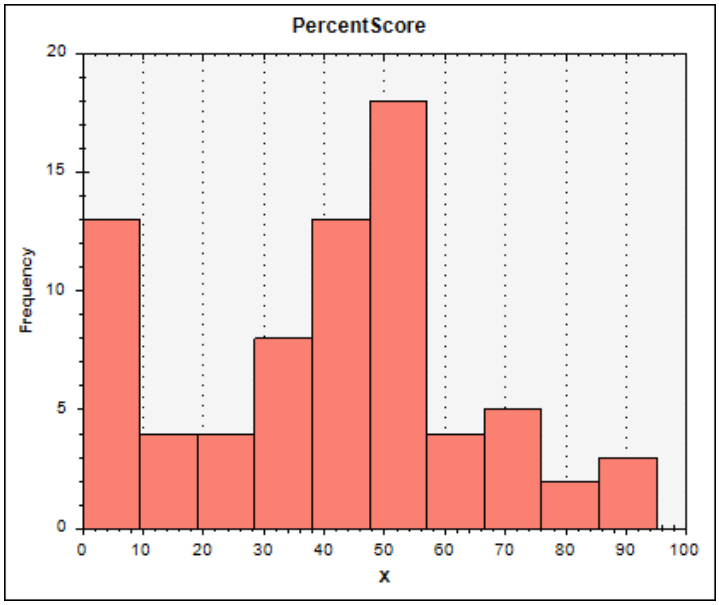

Figure 4.11 – Pretest and Posttest Percent score Levels for Test 2

4.5.2.5 Combined Pretest and Posttest Competence Levels for Test 2

The abilities of all students is shown in Figure 4.12, which illustrates the student's skills were between 0 and 1. This means that the students learning rate had increased and most computer-programming students had gained computer-programming competence. The slope is not so steep, which also indicates that majority of the students skills were good, which is a clear indication that they had gained computer-programming skills.

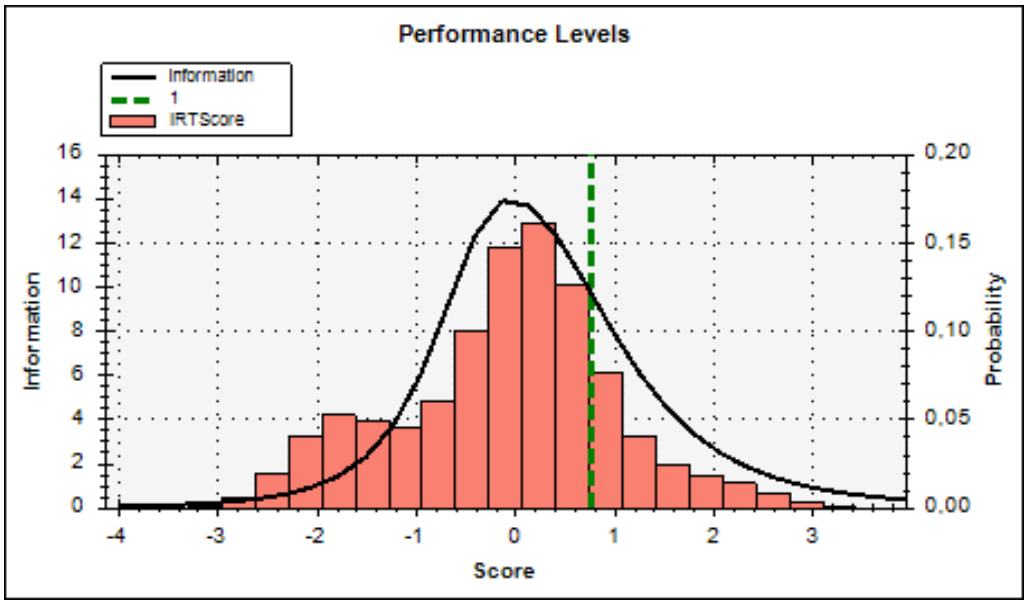

Figure 4.12 – Pretest and Posttest Competence Levels Test 2

4.6 Summary

This chapter discussed the findings obtained from the experimental results. The researcher compared the model derived and formed in Chapter 3 with those models found in the literature, to determine if the blended learning technique had developed student's computer-programming competence. This chapter reported the results for Test 1 pretest and posttest, Test 2 pretest and posttest and the last set of results, which had used the "ANDNOT" logic to combine the results. The Test 1 posttest results were not acceptable, as many questions had generated a result that was out of range, which had resulted in this test not being a good indication of determining whether students had developed computer-programming competence. However, the same results were documented in Test 2, and the results did not determine whether students had developed computer-programming competence. The last set of results, which had combined pretest and posttest data for Test 1 and combined pretest and posttest data for Test 2. These results were a good indication that students had, in fact, developed computer-programming competence. The final chapter will summarise the study, state its main limitations, and present recommendations for further research.

CHAPTER FIVE:
SUMMARY, RECOMMENDATIONS FOR FUTURE RESEARCH AND CONCLUSION

This chapter reflects upon the entire study reported in this dissertation by summarising the work done, highlighting the research gap, and explaining the unique contributions of this study. Moreover, this chapter reports the conclusions and recommendations that resulted from this study for the future study.

5.1 Summary

The overriding aim of this study was to develop a blended learning pedagogical technique in a web-based teaching and learning environment that can practically improve the computer-programming competence of information technology students. To achieve this particular aim, the following objectives were set:

a) To explore how a blended learning pedagogical technique could be implemented in a web-based teaching and learning environment that could help students in designing, understanding, communicating and collaborating to improve their computer-programming competence.

b) To implement a blended learning pedagogical technique in a Blackboard web-based teaching and learning environment to support the development of computer-programming competence of information technology students.

c) To determine whether a blended learning pedagogical technique implemented in a Blackboard web-based teaching and learning environment improves the computer-programming competence of information technology students.

The first objective outlined above focused more extensively on determining whether the blended learning pedagogical technique, together with the web-based teaching and learning environment, had proven to be successful in this research study. The researcher chose the blackboard learning management system, because the researcher's University of is currently using blackboard for all their students to communicate. The blackboard system had been used by academic staff and students on a very small scale, until recently, when the university started driving an e-learning initiative and started to get academics and students to work more and more extensively on blackboard learning management system using all of the latest features and resources available. The relevant information extracted from the literature provided the researcher with the knowledge and understanding of how to explore a blended learning pedagogical technique to help students in designing, understanding, communicating and collaborating within a web supported learning environment so as to improve their computer-programming competence. The outcome of such a web-based environment is the design of the WebTLA framework that serves as a useful guide to implement programming competence development lessons amongst information technology students.

The second research objective was achieved by implementing the WebTLA framework as a blended learning pedagogical technique in the blackboard web-based teaching and learning environment. The practical implementation of this learning framework helped to support the development of computer-programming competence of information technology students. Students were very interested and enthusiastic about using the blackboard environment to facilitate their learning of computer-programming, because it provides the opportunity for them to interact in groups. They created groups within their classrooms to foster online communication and collaboration. They also assisted other students in understanding and communicating within the blackboard environment. Lecturers had used blackboard environment to upload learning resources that students could access at any time. There were announcements that lecturers had used to notify students of tests and examinations taking place. There were a whole host of features and resources available for students to use with which to familiarise themselves. Therefore, this learning management system was the ideal system to utilise in order to implement this research study.

It was important to determine whether a blended learning pedagogical technique implemented in a blackboard web-based teaching and learning environment improves the computer-programming competence of information technology students as the last study objective. Most

students who had entered the university had very little or no prior knowledge about computers. Therefore, using the online blackboard environment together with the blended learning technique, students had shown positive signs that their computer-programming competence was developed. The web-based environment had supported the students to understand and learn computer-programming. This study's findings revealed that students had no computer-programming competence, when they had started the computer-programming course at DUT, although most students had completed their secondary education. Many of these students had very little or no prior computer-programming knowledge, which had impacted on their understanding and comprehension of the computer-programming concepts being taught in the classroom, and they became more comfortable with the environment as the semester progressed. The approach used in this research study had supported the students so that they had access to infrastructure, resources, their peers and fellow students, so as to assist and develop their computer-programming competence using the WebTLA framework towards computer-programming competence.

The retrospective pretest approach had been very successful in this research study to find evidence that would create a link between students' prior knowledge of computer-programming and their actual computer-programming competence. The empirical results of this study obtained using the item response theory had shown that retrospective pretest results for each test was reported to be very bad, as some questions were identified to have generated a value that was out of range. The researcher had combined the pretest and posttest results using the "ANDNOT" logic gate approach for each test, and the actual results were good. Students that claimed to not have any knowledge of computer-programming prior to starting the course and thereafter solving the question (item) correctly, had, in fact, developed computer-programming competence during the semester, because there was a positive change in their behaviour. The technique of retrospective pretest design with item response theory had proved to be successful in determining whether students had developed their computer-programming competence during an intervention process. The technique does not allow students to overestimate or underestimate their computer-programming competence, by eliminating any effects arising from overestimation or underestimation. Therefore the results produced were considered reliable.

5.2 Recommendations for Future Research

There are many changes to technology each year, and one recommendation for further research is to utilise similar WebTLA framework to determine whether other universities across the country can adopt and use it for improving student's computer-programming competence. Computer-programming is a diverse field, and it will never remain static, therefore universities need to understand that they need to try and incorporate different types of web-based technologies into their curricula so as to ensure that their failure rates are decreasing. Consequently, it would be prudent to consider validating the WebTLA framework across universities and in different programmes to establish its versatility and generality. Many universities are still using the traditional approach to teaching and learning; they need to research new and innovative approaches to teaching and learning at universities. Universities need to also understand that many students have not even seen a computer before entering a computer-programming class, therefore they need to be schooled around the latest and most modern technology, by introducing some orientation of computers. It may be better for any researcher to avoid using online retrospective surveys, as most respondents did not complete the questionnaire online.

The WebTLA framework can be used by other university as a useful guide to create their own frameworks to use in their classes. Moreover, many universities can adopt this framework and prove this framework to have been successful, not only at DUT, but at other universities as well. These findings have contributed towards some valuable insights into teaching and learning in higher education at the DUT in South Africa. These findings further highlight using a different and unique approach to teaching and learning in a web-based blended teaching and learning environment that has helped students to develop their computer-programming competence. Contributions and future recommendations were made based on these findings. Through an extensive review of the literature, this study has shown the importance for further research based on these findings. It is interesting to note that this study is putting forward a novel evaluation framework that examines assessment from both student and lecturer perspectives. Currently, the education practice of assessment of student knowledge tends to ignore the opinions of students in assessment and bias towards lecturers. When students don't do well in a subject, they take absolute blame, and lecturers are always right. This study has shown that it is not always right for students to take blame for their failures. A particular item might not be appropriate for a test, where questions might be too difficult, and students might not be learning something different

from what they know. It would be prudent to conduct more research on the usefulness of embedding the opinions of students into learning assessments.

5.3 Conclusion

In conclusion, studies like this, which are aimed at using a unique blended learning teaching and learning techniques incorporating web-based learning managements systems, had immensely benefitted the information technology students at the Durban University of Technology. It is recommended to conduct further research at more universities, as well as to consider such research as an on-going process that can innovate student assessment.

This dissertation had tested information technology students' knowledge about computers and computer-programming prior to starting the computer-programming course. There is a relationship that exists between the student's prior knowledge of computers and their computer-programming competence. The current research study had utilised a web-based teaching and learning environment to test and develop computer-programming competence of information technology students. This research study was successful in developing computer-programming competence of students during the semester computer-programming course.

BIBLIOGRAPHY

Abdulghani, H. M., Shaik, S. A., Khamis, N., Al-Drees, A. A., Irshad, M., Khalil, M. S., Alhaqwi, A. I. and Isnani, A. 2014. Research methodology workshops evaluation using the Kirkpatrick's Model: translating theory into practice. Medical teacher, 36 (S1): S24-S29.

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. 2002. Agile software development methods: Review and analysis: VTT Finland.

Acar, T. 2015. An Investigation of Agreement Between the Item Difficulty Coefficient Calculated in Accordance With Classical Test Theory and Item Response Theory With Bland–Altman Method. Communications in Statistics-Theory and Methods, 44 (21): 4614-4621.

Agarwal, P. and Hundhausen, C. D. 2010. A socio-psychological approach to improve student participation and review quality in peer code reviews. In: Proceedings of 2010 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). Leganes, 21-25 September 2010. IEEE, 263-264.

Altintas, T., Gunes, A. and Sayan, H. 2014. A peer-assisted learning experience in computer programming language learning and developing computer programming skills. Innovations in Education and Teaching International, (ahead-of-print): 1-9.

Aluko, F. R. and Shonubi, O. K. 2014. Going beyond Kirkpatrick's Training Evaluation Model: the role of workplace factors in distance learning transfer. Africa Education Review, 11 (4): 638-657.

Atef, H. and Medhat, M. 2015. Blended Learning Possibilities in Enhancing Education, Training and Development in Developing Countries: A Case Study in Graphic Design Courses. Technology, Education, Management, Informatics Journal, 4 (4): 358-365.

Atif, Y. 2013. Conversational learning integration in technology enhanced classrooms. Computers in Human Behavior, 29 (2): 416-423.

Ayob, A., Bais, B., Abd Aziz, N., Arsad, N. and Husain, H. 2011. Use of Rasch Analysis in engineering students psychometric evaluation. In: Proceedings of 2011 3rd International Congress on Engineering Education (ICEED), . Kuala Lumpur, 7-8 December 2011. IEEE, 214-217.

Azevedo, R., Witherspoon, A., Graesser, A., McNamara, D., Rus, V., Cai, Z., Lintean, M. and Siler, E. 2008. MetaTutor: An adaptive hypermedia system for training and fostering self-regulated learning about complex science topics. In: Proceedings of annual meeting of the Society for Computers in Psychology, Chicago, IL.

Ballantine, J. A., Larres, P. M. and Oyelere, P. 2007. Computer usage and the validity of self-assessed computer competence among first-year business students. Computers & Education, 49 (4): 976-990.

Barchino, R., Gutiérrez, J. M., De-Marcos, L., Martínez, J. J., Jiménez, L., Otón, S., Gutiérrez, J. A. and Hilera, J. R. 2012. Experiences in the use of Mobile Games to improve Programming Skills in Computer Engineering. International Journal of Innovative Computing, Information and Control, 8 (2): 1167-1174.

Barik, N., Jena, P. and Sethy, N. 2014. Assessing Information Communication Technology (ICT) skills of degree science students of an autonomous college of Odisha: a survey. Paper presented at the 1st International Virtual Conference on Information Retrieval on Scienific Literature: Emerging Frontiers and Challenges (Online). Mullana-Ambala, Haryana, 10-11 June 2014. Available: http://eprints.rclis.org/24974/ (Accessed

Bennedsen, J. and Caspersen, M. E. 2007. Failure rates in introductory programming. New York, NY, USA: 26 March 2014).

Berry, M. and Kölling, M. 2013. The design and implementation of a notional machine for teaching introductory programming. In: Proceedings of the 8th Workshop in Primary and Secondary Computing Education. Aarhus, Denmark, November 11-13 2013. ACM, 25-28. Available: http://kar.kent.ac.uk/37645/1/2013-09-WiPSCE-Notional-Machine.pdf (Accessed 3 April 2014).

Blackie, M. A., Case, J. M. and Jawitz, J. 2010. Student-centredness: The link between transforming students and transforming ourselves. Teaching in Higher Education, 15 (6): 637-646.

Boughey, C. 2013. Real barrier in academic inertia. Available: http://mg.co.za/article/2013-10-04-real-barrier-is-academic-inertia (Accessed 3 April 2014).

Brito, M. A. and de Sá-Soares, F. 2014. Assessment frequency in introductory computer programming disciplines. Computers in Human Behavior, 30: 623-628.

Brown, S. 2004. Assessment for learning. Learning and teaching in higher education, 1 (1): 81-89.

Burgess, G., Grogan, S. and Burwitz, L. 2006. Effects of a 6-week aerobic dance intervention on body image and physical self-perceptions in adolescent girls. Elsevier, 3 (1): 57-66.

Butler, M. and Morgan, M. 2007. Learning challenges faced by novice programming students studying high level and low feedback concepts. Proceedings ascilite Singapore: 99-107.

Campbell, L. M. 2008. Beginning with what students know: the role of prior knowledge in learning. In: Mindful learning: 101 proven strategies for student and teacher success. Corwin Press Inc, 192.

Cappelleri, J. C., Lundy, J. J. and Hays, R. D. 2014. Overview of classical test theory and item response theory for the quantitative assessment of items in developing patient-reported outcomes measures. Clinical therapeutics, 36 (5): 648-662.

Carless, D. 2014. Exploring learning-oriented assessment processes. Higher Education, 69 (6): 963-976.

Carter, A. S. and Hundhausen, C. D. 2011. A review of studio-based learning in computer science. Journal of Computing Sciences in Colleges, 27 (1): 105-111.

Cartwright, F. 2013. Item and Test Analysis Available: http://polymetrika.com/home/IATA

Cha, S. E., Jun, S. J., Kwon, D. Y., Kim, H. S., Kim, S. B., Kim, J. M., Kim, Y., Han, S. G., Seo, S. S. and Jun, W. C. 2011. Measuring achievement of ICT competency for students in Korea. Computers & Education, 56 (4): 990-1002.

Chen, J., Su, H.-M. and Liu, J.-H. 2007. A curriculum design on embedded system education for first-year graduate students. In: Proceedings of Parallel and Distributed Systems, 2007 International Conference on. Hsinchu, IEEE, 1-6.

Chen, J. J.-Y. and Wu, M. M.-Z. 2015. Integrating extreme programming with software engineering education. In: Proceedings of Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on. IEEE, 577-582.

Chen, S. Y. and Huang, P.-R. 2013. The comparisons of the influences of prior knowledge on two game-based learning systems. Computers & Education, 68: 177-186.

Chrysafiadi, K. and Virvou, M. 2012. Evaluating the integration of fuzzy logic into the student model of a web-based learning environment. Expert Systems with Applications, 39 (18): 13127-13134.

Claro, M., Preiss, D. D., San Martín, E., Jara, I., Hinostroza, J. E., Valenzuela, S., Cortes, F. and Nussbaum, M. 2012. Assessment of 21st century ICT skills in Chile: test design and results from high school level students. Computers & Education, 59 (3): 1042-1053.

Connolly, C., Murphy, E. and Moore, S. 2009. Programming anxiety amongst computing students—a key in the retention debate? IEEE Transactions on Education,, 52 (1): 52-56.

Cordova, J. R., Sinatra, G. M., Jones, S. H., Taasoobshirazi, G. and Lombardi, D. 2014. Confidence in prior knowledge, self-efficacy, interest and prior knowledge: influences on conceptual change. Contemporary Educational Psychology, 39 (2): 164-174.

Corney, M., Teague, D. and Thomas, R. N. 2010. Engaging students in programming. In: Proceedings of the Twelfth Australasian Conference on Computing Education-Volume 103. Brisbane, Australia, Australian Computer Society, Inc., 63-72.

Culligan, B. 2011. Item Response Theory, reliability and standard error. 23 September 2015).

Culpepper, S. A. 2015. Revisiting the 4-parameter item response model: Bayesian estimation and application. Psychometrika: 1-22.

Cutler, S. and Borrego, M. 2010. Developing global competence in graduate engineering and science students through an IGERT international internship program. In: Proceedings of Frontiers in Education Conference (FIE), 2010 IEEE. Washington, DC, IEEE, F3H-1-F3H-6.26 March 2014).

Cutrer, W. B., Sullivan, W. M. and Fleming, A. E. 2013. Educational strategies for improving clinical reasoning. Current problems in pediatric and adolescent health care, 43 (9): 248.

Danner, R. and Pessu, C. 2013. A survey of ICT competencies among students in teacher preparation programmes at the University of Benin, Benin City, Nigeria. Journal of Information Technology Education: Research, 12 (1): 33-49.

Dellwo, D. R. 2010. Course assessment using multi-stage pre/post testing and the components of normalized change. Journal of the Scholarship of Teaching and Learning, 10 (1): 55-67.

Dimitrov, D. M. and Rumrill Jr, P. D. 2003. Pretest-posttest designs and measurement of change. Work, 20 (2): 159-165.

Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N. B. 2012. A decade of agile methodologies: Towards explaining agile software development: Elsevier.

Dreyer, A., Couper, I., Bailey, R., Talib, Z., Ross, H. and Sagay, A. 2015. Identifying approaches and tools for evaluating community-based medical education programmes in Africa. African Journal of Health Professions Education, 7 (1): 134-139.

Emmanuel, A. K., Oluwadamilare, O. G. and Yomi, O. A. 2015. Improvement Strategies for Computer Science Students' Academic Performance in Programming Skill. American Journal of Computer Science and Information Engineering, 2 (5): 45-50.

Fearon, C., McLaughlin, H. and Yoke Eng, T. 2012. Using student group work in higher education to emulate professional communities of practice. Education+ Training, 54 (2/3): 114-125.

Fernandez, C. S., Noble, C. C., Jensen, E. and Steffen, D. 2015. Moving the needle: a retrospective pre-and post-analysis of improving perceived abilities across 20 leadership skills. Maternal and child health journal, 19 (2): 343-352.

Fessakis, G., Gouli, E. and Mavroudi, E. 2013. Problem solving by 5-6 years old kindergarten children in a computer programming environment: a case study. Computers & Education, 63: 87-97.

Fischer, R. 2011. Cross-cultural training effects on cultural essentialism beliefs and cultural intelligence. International Journal of Intercultural Relations, 35 (6): 767-775.

Foley, B. P. 2010. Improving IRT parameter estimates with small sample sizes: Evaluating the efficacy of a new data augmentation technique. Phd, University of Nebraska - Lincoln.

Forehand, M. 2010. Bloom's taxonomy. Emerging perspectives on learning, teaching, and technology: 41-47.

Gasparinatou, A. and Grigoriadou, M. 2015. Supporting Student Learning in Computer Science Education via the Adaptive Learning Environment ALMA. Systems, 3 (4): 237-263.

Ginns, P. and Ellis, R. 2007. Quality in blended learning: Exploring the relationships between on-line and face-to-face teaching and learning. The Internet and Higher Education, 10 (1): 53-64.

Govender, D. W., Govender, I., Breed, B., Havenga, M., Mentz, E., Dignum, F. and Dignum, V. 2013. Supporting information technology teachers through programming professional development: a South African case study. Journal of Communication, 4 (2): 153-160.

Graziano, A. M. and Raulin, M. L. 2013. Research methods: a process of inquiry. Boston, Mass: Pearson.

Hakkarainen, K., Ilomäki, L., Lipponen, L., Muukkonen, H., Rahikainen, M., Tuominen, T., Lakkala, M. and Lehtinen, E. 2000. Students' skills and practices of using ICT: Results of a national assessment in Finland. Computers & Education, 34 (2): 103-117.

Hare, B. K. 2013. Classroom interventions to reduce failure & withdrawal in CS1: a field report. Journal of Computing Sciences in Colleges, 28 (5): 228-235.

Harvie, D. P. and Agah, A. 2016. Targeted Scrum: Applying Mission Command to Agile Software Development. IEEE Transactions on Software Engineering, 42 (5): 476-489.

Hatakka, M., Andersson, A. and Grönlund, Å. 2013. Students' use of one to one laptops: a capability approach analysis. Information Technology & People, 26 (1): 94-112.

Hooshyar, D., Ahmad, R. B., Shamshirband, S., Yousefi, M. and Horng, S.-J. 2015. A flowchart-based programming environment for improving problem solving skills of Cs minors in computer programming. The Asian International Journal of Life Sciences, 24 (2): 629-646.

Hsieh, T.-C., Lee, M.-C. and Su, C.-Y. 2013. Designing and implementing a personalized remedial learning system for enhancing the programming learning. Journal of Educational Technology & Society, 16 (4): 32-46.

Ilahi, M., Belcadhi, L. C. and Braham, R. 2013. Towards a competence web -based assessment model to support lifelong learning. In: Proceedings of 2013 Fourth International Conference on Information and Communication Technology and Accessibility (ICTA). Hammamet 24-26 October 2013. IEEE, 1-6.

Isong, B. 2014. A methodology for teaching computer programming: first year students' perspective. International Journal of Modern Education and Computer Science, 6 (9): 15-21.

Ivančević, V. 2014. Constructing programming tests from an item pool: pushing the limits of student knowledge using assessment and learning analytics. Journal of Learning Analytics, 1 (3): 161-164.

Judson, E. 2012. Learning about bones at a science museum: examining the alternate hypotheses of ceiling effect and prior knowledge. Instructional Science, 40 (6): 957-973.

Junker, B. W. 2012. Some aspects of classical reliability theory & classical test theory.

Jurado, F., Redondo, M. and Ortega, M. 2014. eLearning standards and automatic assessment in a distributed eclipse based environment for learning computer programming. Computer Applications in Engineering Education, 22 (4): 774-787.

Kabassi, K., Dragonas, I., Ntouzevits, A., Pomonis, T., Papastathopoulos, G. and Vozaitis, Y. 2016. Evaluating a learning management system for blended learning in Greek higher education. SpringerPlus, 5 (1): 1.

Kreth, J., Merritt, J., Shi, W. and Qi, F. 2005. Co-ordinated bacteriocin production and competence development: a possible mechanism for taking up DNA from neighbouring species. Molecular microbiology, 57 (2): 392-404.

Kuk, K., Milentijevic, I., Rancic, D. and Spalevic, P. 2014. Designing intelligent agent in multilevel game-based modules for E-Learning Computer Science course. In: Ivanovic, M. and Jain, L. C. eds. E-Learning paradigms and applications. Heidelberg: Springer, 39-62.

Kuo, C.-Y. and Wu, H.-K. 2013. Toward an integrated model for designing assessment systems: an analysis of the current status of computer-based assessments in science. Computers & Education, 68: 388-403.

Law, K. M., Lee, V. and Yu, Y.-T. 2010. Learning motivation in e-learning facilitated computer programming courses. Computers & Education, 55 (1): 218-228.

Lin, Y.-C., Chung, P., Yeh, R. C. and Chen, Y.-C. 2016. An Empirical Study of College Students' Learning Satisfaction and Continuance Intention to Stick with a Blended e-Learning Environment. International Journal of Emerging Technologies in Learning, 11 (2)

MacDonald, J. and Creanor, L. 2010. Learning with online and mobile technologies: a student survival guide. Burlington, Farnham, England: Gower.

Malliari, A., Korobili, S. and Togia, A. 2012. IT self-efficacy and computer competence of LIS students. Electronic Library, The, 30 (5): 608-622.

Mallinckrodt, B., Miles, J. R. and Recabarren, D. A. 2015. Using focus groups and Rasch item response theory to improve instrument development. The Counseling Psychologist: 1-47.

Mathews, J. M. 2010. Using a studio-based pedagogy to engage students in the design of mobile-based media. English Teaching: Practice & Critique, 9 (1)

Merrouch, F., Hnida, M., Idrissi, M. K. and Bennani, S. 2014. Online placement test based on item response theory and IMS Global standards. IJCSI International Journal of Computer Science, 11 (5)

Mhashi, M. and Alakeel, A. 2013. Difficulties facing students in learning computer programming skills at Tabuk University. In: Proceedings of Proceedings of the 12th International Conference on Education and Educational Technology (EDU'13), Iwate, Japan. 15-24.

Miller, G. E. 1990. The assessment of clinical skills/competence/performance. Academic medicine, 65 (9): S63-67.

Ming-der Wu, S.-T. Y. 2012. Effects of undergraduate student computer competence on usage of library electronic collections. Journal of Library and Information Studies, 10 (1): 1-17.

Mohammed, P. and Mohan, P. 2010. Combining digital games with culture: a novel approach towards boosting student interest and skill development in computer science programming. In: Proceedings of Mobile, Hybrid, and On-Line Learning, 2010. ELML'10. Second International Conference on. IEEE, 60-65.

Ngan, S.-C. and Law, K. M. 2015. Exploratory network analysis of learning motivation factors in e-learning facilitated computer programming courses. The Asia-Pacific Education Researcher: 1-13.

Nielsen, R. B. 2011. A retrospective pretest-posttest evaluation of a one-time personal finance training. Journal of Extension, 49 (1): n1.

Nimon, K. 2013. Explaining differences between retrospective and traditional pretest self-assessments: competing theories and empirical evidence. International Journal of Research & Method in Education, 37 (3): 256-269.

Olugbara, O. O., Millham, R., Heukelman, D., Thakur, S., Wesso, H. W. and Sharif, M. 2014. Determining E-skills interventions to improve the effectiveness of service delivery by community development workers.

Pérez, J. and Murray, M. C. 2010. Generativity: the new frontier for information and communication technology literacy.

Petrillo, J., Cano, S. J., McLeod, L. D. and Coon, C. D. 2015. Using Classical Test Theory, Item Response Theory, and Rasch Measurement Theory to evaluate patient-reported outcome measures: a comparison of worked examples. Value in Health, 18 (1): 25-34.

Porter, L. and Simon, B. 2013. Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In: Proceedings of Proceeding of the 44th ACM technical symposium on Computer science education. Denver, Colorado, USA, 6-9 March 2013. ACM, 165-170.

Rane-Sharma, A., Sharma, C., Raman, R. and Sasikumar, M. 2010. A methodology for enhancing programming competence of students using Parikshak. In: Proceedings of Technology for Education (T4E), 2010 International Conference on. IEEE, 24-31.

Ranjeeth, S. and Naidoo, R. 2011. An investigation into the relationship between the level of cognitive maturity and the types of errors made by students in a computer programming course. College Teaching Methods & Styles Journal (CTMS), 3 (2): 31-40.

Reardon, S. and Tangney, B. 2014. Smartphones, studio-based learning, and scaffolding: helping novices learn to program. ACM Transactions on Computing Education, 14 (4): 1-15.

Rosli, M. M., Ibrahim Teo, N. H. and Khairol Azmi, N. N. 2010. Undergraduate student preference activities and sources to learn programming. In: Proceedings of Engineering Education (ICEED), 2010 2nd International Congress on. IEEE, 47-51.26 March 2014).

Rust, C. 2002. The impact of assessment on student learning how can the research literature practically help to inform the development of departmental assessment strategies and learner-centred assessment practices? Active learning in higher education, 3 (2): 145-158.

Savignon, S. J. 1976. Communicative Competence: Theory and Classroom Practice. Paper presented at the Central States Conference on Teaching of Foreign Languages. Detroit, 23 April 1976. ERIC,

Schumm, M., Joseph, S., Schroll-Decker, I., Niemetz, M. and Mottok, J. 2012. Required competences in software engineering: pair programming as an instrument for facilitating life-long learning. In: Proceedings of Interactive Collaborative Learning (ICL), 2012 15th International Conference on. IEEE, 1-5.

Serrano-Cámara, L. M., Paredes-Velasco, M., Alcover, C.-M. and Velazquez-Iturbide, J. Á. 2014. An evaluation of students' motivation in computer-supported collaborative learning of programming concepts. Computers in Human Behavior, 31: 499-508.

Shannon, L. Y. and Bennett, J. 2012. A case study: applying critical thinking skills to computer science and technology. Information Systems Education Journal, 10 (4): 41.

Shantakumari, N. and Sajith, P. 2015. Blended learning: the student viewpoint. Annals of medical and health sciences research, 5 (5): 323-328.

Srikant, S. and Aggarwal, V. 2014. A system to grade computer programming skills using machine learning. In: Proceedings of Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. New York, 24-27 August 2014. ACM, 1887-1896.

Svinicki, M. 1993. What they don't know can hurt them: the role of prior knowledge in learning. 1 September 2015).

Taub, M., Azevedo, R., Bouchet, F. and Khosravifar, B. 2014. Can the use of cognitive and metacognitive self-regulated learning strategies be predicted by learners' levels of prior knowledge in hypermedia-learning environments? Computers in Human Behavior, 39: 356-367.

Tavakol, M., Rahimi-Madiseh, M. and Dennick, R. 2014. Postexamination Analysis of Objective Tests Using the Three-Parameter Item Response Theory. Journal of nursing measurement, 22 (1): 94-105.

Templin, J. 2016. Jonathan Templins Website. Available: http://www.google.co.za/url?sa=t&rct=j&q=&esrc=s&source=web&cd=6&cad=rja&uact=8&ved=0ahUKEwierJHcud7NAhUMCsAKHWUSA7EQFgg8MAU&url=http%3A%2F%2Fjonathantemplin.com%2Ffiles%2Firt%2Firt11icpsr%2Firt11icpsr_lecture02.pdf&usg=AFQjCNGAneM2BU0HE74C5DxJfs98onLJZA&bvm=bv.126130881,d.d24

Trochim, W. M., Donnelly, J. P. and Arora, K. 2014. Research Methods: the essential knowledge base. Canada: Cengage Learning.

Unterkalmsteiner, M., Gorschek, T., Islam, A. K. M. M., Cheng, C. K., Permadi, R. B., Feldt, R., Blekinge Tekniska, H. and Sektionen för datavetenskap och, k. 2012. Evaluation and Measurement of Software Process Improvement-A Systematic Literature Review. IEEE Transactions on Software Engineering, 38 (2): 398-424.

Van der Vleuten, C., Schuwirth, L., Scheele, F., Driessen, E. and Hodges, B. 2010. The assessment of professional competence: building blocks for theory development. Best Practice & Research Clinical Obstetrics & Gynaecology, 24 (6): 703-719.

Van Vliet, P. J., Kletke, M. G. and Chakraborty, G. 1994. The measurement of computer literacy: a comparison of self-appraisal and objective tests. International Journal of Human-Computer Studies, 40 (5): 835-857.

Veas, A., Gilar, R., Miñano, P. and Castejón, J.-L. 2016. Estimation of the Proportion of Underachieving Students in Compulsory Secondary Education in Spain: An Application of the Rasch Model. Frontiers in psychology, 7

Vivian, R., Falkner, K. and Falkner, N. 2013. Computer science students' causal attributions for successful and unsuccessful outcomes in programming assignments. Paper presented at the the 13th Koli Calling International Conference on Computing Education Research. Koli, Finland, ACM, 125-134. 26 August 2015).

Wang, T., Su, X., Ma, P., Wang, Y. and Wang, K. 2011. Ability-training-oriented automated assessment in introductory programming course. Computers & Education, 56 (1): 220-226.

Wanner, T. and Palmer, E. 2015. Personalising learning: Exploring student and teacher perceptions about flexible learning and assessment in a flipped university course. Computers & Education, 88: 354-369.

Wood, S., Michaelides, G. and Thomson, C. 2013. Successful extreme programming: Fidelity to the methodology or good teamworking? Information and Software Technology, 55 (4): 660-672.

Yang, T.-C., Hwang, G.-J., Yang, S. J. and Hwang, G.-H. 2015. A two-tier test-based approach to improving students' computer-programming skills in a web-based learning environment. Journal of Educational Technology & Society, 18 (1): 198-210.

Zacharis, N. Z. 2012. Predicting college students' acceptance of podcasting as a learning tool. Interactive Technology and Smart Education, 9 (3): 171-183.

Zhong, X., Madhavji, N. H. and El Emam, K. 2000. Critical factors affecting personal software processes. IEEE software, 17 (6): 76-83.

Zottmann, J. M., Stegmann, K., Strijbos, J.-W., Vogel, F., Wecker, C. and Fischer, F. 2013. Computer-supported collaborative learning with digital video cases in teacher education: the impact of teaching experience on knowledge convergence. Computers in Human Behavior, 29 (5): 2100-2108.

Appendix

LETTER OF INFORMATION

Dear Respondent

I am a student registered at the Durban University of Technology in the Department of Information Technology currently pursuing a Masters Degree in Information Technology and the primary component of this degree deals with a research-based investigation that involves student feedback as well as data collection. I will be grateful if you could please complete the attached questionaire. This questionaire should take approximately 15 minutes to complete and you can be assured that your response will receive the utmost confidentiality and any information will not be divulged to any other person. The researcher and supervisor of this study will only have access to this information. Your co-operation in assisting me with this vital component of my study is highly appreciated and I take this opportunity of thanking you in advance for enabling me to complete this research project. Your assistance is greatly appreciated.

Title of the Research Study: Developing a web-based technique to improve computer-programming competence of Information Technology students

Principal Researcher: P Jackson

Supervisor: Prof O.O. Olugbara - PHD Computer Science

Brief Introduction and Purpose of the Study: This study proposes to explore a web-based studio technique with minimal instructor intervention to improve the computer-programming competence of information technology students. The technique being proposed in this study will be implemented in a blackboard web-based environment to help develop computer competence of information technology students and to evaluate the effectiveness of the technique. Analysis of the results will reveal whether the developed technique has a positive impact on computer-programming competence of information technology students. This research study will allow information technology students to dynamically collaborate with their peers with minimal instructor intervention towards improving their computer-programming competence.

Risks or Discomforts to the Participant: No risk or discomfort to participant.

Persons to Contact in the Event of Any Problems or Queries:

Supervisor: Professor O.O Olugbara, PHD Computer Science – 031373 5591

Researcher: Mrs. P. Jackson – 0845142826 or 031373 5579 or the Institutional Research Ethics administrator on 031 373 2900. Complaints can be reported to the DVC: TIP, Prof F. Otieno on 031 373 2382 or dvctip@dut.ac.za.

General: Participation is voluntary and the approximate number of participants are between 1 to 120 students.

CONSENT

Statement of Agreement to Participate in the Research Study:

- I hereby confirm that I have been informed by the researcher, P. Perumal Jackson, about the nature, conduct, benefits and risks of this study - Research Ethics Clearance Number: REC 13/15,
- I have also received, read and understood the above written information (Participant Letter of Information) regarding the study.
- I am aware that the results of the study, including personal details regarding my sex, age, date of birth, initials and diagnosis will be anonymously processed into a study report.
- In view of the requirements of research, I agree that the data collected during this study can be processed in a computerised system by the researcher.
- I may, at any stage, without prejudice, withdraw my consent and participation in the study.
- **I have had sufficient opportunity to ask questions and (of my own free will) declare myself prepared to participate in the study.**
- **I understand that significant new findings developed during the course of this research, which may relate to my participation will be made available to me.**

_____

Signature / Right Thumbprint

I, Priyalushinee Perumal Jackson herewith confirm that the above participant has been fully informed about the nature, conduct and risks of the above study.

Priyalushinee Jackson _____

Full Name of Researcher **Signature**

Group 1 – Test2 – C++ – Pretest and Posttest

In the program below, answer questions about the program.

```
#ifndef Laundry_H
#define Laundry_H
#include<string>

class Laundry
{
  public : Laundry();          Laundry(string, string, int);
         void setLaundryName(string);
         void setLaundryType(string);
         void setLaundryWeight(int);
         void setdetails(string,string,int);

         string getLaundryName();
         string getLaundryType();

         int getLaundryWeight();

         double BasicCost();

              void display();
                        ~Laundry();

private :   string Name;
            string type;
            float weight;
             };
             #endif
```

In the program below, there are snippets of code missing from this program, please fill in

the blanks.

	Line
#include <iostream>	1
using namespace std;	2
#include<string>	3
#include "Laundry.h"	4
Laundry:: __**1.**_	5
{	6
Name=" ";	7
weight=__**2.**__;	8
type=' ';	9
}	10
Laundry::Laundry(string	
__**3.**__, string typ, float wei)	11
{	12
setLaundryName(name);	13
setLaundryType(typ);	14
setLaundryWeight(wei);	15
}	16
void Laundry::__**4.**__(string n)	17
{	18
Name=n;	19
}	20
string Laundry::getLaundryName()	21
{	22
return Name;	23
}	24
void Laundry::setLaundryType(string t)	25
{	26

```
    type=t;                                  27
}                                            28

string Laundry::getLaundryType()             29
{                                            30
      return __5.__;                         31
}                                            32

void Laundry::__6.__(float w)                33
{                                            34
       weight=__7.__;                        35
}                                            36
float Laundry::getLaundryWeight()            37
{                                            38
      return weight;                         39
}                                            40

__8.__ Laundry::BasicCost()                  41
{                                            42
   return (__9.__* 100);                     43
}                                            44

void Laundry::display()                      45
{                                            46
                                             Line
cout<<"\nName  : "<<(
       *this).getLaundryName()<<"\n";        47
  cout<<"\nWeight:"<<. (*this).
getLaundryWeight()<<"\n";                    48
cout<<"\nCost is :"<<
this.BasicCost()<<"\n\n";                    49
}                                            50
```

```
void Laundry::setdetails(string n,
string t,float c)                    51
{                                    52
  setLaundryName(n);                 53
    setLaundryType(t);               54
    setLaundryWeight(c);             55
 }                                   56

Laundry:: ~Laundry()                 57
{                                    58
        cout<<"\nObject deleted ! ";  59
}                                    60
```

1. What is your gender?	☐ Male ☐ Female	
2. What is your age?	☐ 18-25 ☐ 26-33 ☐ 34-39 ☐ 40 or older	
3. Which of the following describes the area you live in?	☐ Urban ☐ Rural	
4. What is your primary language?	☐ English ☐ Afrikaans ☐ Zulu ☐ Other	
5. Are you currently staying at the DUT residence?	☐ Yes ☐ No	
	Before taking the Programming course, I can solve the following problems.	**After taking the Programming course, I can solve the following problems.**
6. In the above segment of code, what is the class name?	☐ Yes ☐ No	☐ **Laundry** ☐ Class Laundry ☐ ClassLaundry ☐ None of the above
	Before taking the	**After taking the Programming**

	Programming course, I can solve the following problems.	**course, I can solve the following problems.**
7. In the above segment of code, which line is the default constructor header located?	☐ Yes ☐ No	☐ Line 1 ☐ **Line 5** ☐ Line 25 ☐ Line 31
8. In the above segment of code, which line is the overloaded constructor header located?	☐ Yes ☐ No	☐ Line 7 ☐ **Line 11** ☐ Line 19 ☐ Line 20
9. In the above segment of code, what is the header name of the mutator method that sets the laundry name?	☐ Yes ☐ No	☐ **setLaundryName(string)** ☐ setLaundNAme() ☐ setName() ☐ setLaundryName1
10. In the above segment of code, which of the following is not a header for the accessor method?	☐ Yes ☐ No	☐ void getLaundryName( ) ☐ void getLaundryType( ) ☐ void getLaundryWeight( ) ☐ **double BasicCost( )**
11. In the above segment of code, which line shows the header for the destructor?	☐ Yes ☐ No	☐ Line 1 ☐ **Line 53** ☐ Line 57 ☐ None of the above
12. In the above segment of code, what are the names of the private data members?	☐ Yes ☐ No	☐ **Name, type,weight** ☐ Nam,typ,wei ☐ N,t,w ☐ NAME,TYPE,WEIGHT
	Before taking the	**After taking the Programming**

	Programming course, I can solve the following problems.	**course, I can solve the following problems.**
13. What is the missing code for blank number 1?	☐ Yes ☐ No	☐ **Laundry()** ☐ Laundry ☐ laundry ☐ laundry()
14. What is the missing code for blank number 2?	☐ Yes ☐ No	☐ **0.0** ☐ LaundryWeight ☐ 10 ☐ 100
15. What is the missing code for blank number 3?	☐ Yes ☐ No	☐ **name** ☐ name( ) ☐ Name( ) ☐ NAMe( )
16. What is the missing code for blank number 4?	☐ Yes ☐ No	☐ **setLaundryName** ☐ LaundryName() ☐ laundryname ☐ laundry_name
17. What is the missing code for blank number 5?	☐ Yes ☐ No	☐ LaundryType ☐ Laundrytype( ) ☐ Laundrytype ☐ **type**
18. What is the missing code for blank number 6?	☐ Yes ☐ No	☐ weight ☐ **setLaundryWeight** ☐ propLaundryWeight1 ☐ LaundryWeight
19. What is the missing code for blank number 7?	☐ Yes ☐ No	☐ value ☐ **w** ☐ weight ☐ Weight
	Before taking the	**After taking the Programming**

	Programming course, I can solve the following problems.	**course, I can solve the following problems.**
20. What is the missing code for blank number 8?	☐ Yes ☐ No	☐ integer ☐ char ☐ **double** ☐ string
21. What is the missing code for blank number 9?	☐ Yes ☐ No	☐ proplaundryeight ☐ LaundryWeight ☐ propLaundryWeight1 ☐ **weight**

In the program below, there are snippets of code missing from this program, please fill in the blanks.

```
int main ()
{
Laundry __10.__("smith","A",10);
obj1.__11.__();
string name, type;
int weight;
   cout<<"_____________";
cout<<"\nPlease enter name ";
cin>>name;
cout<<"\nPlease type of laundry ";
cin>>type;
cout<<"\nPlease enter the weight of laundry";
cin>>weight;
cout<<"_____________";
                __12.__ obj2 (name,type,weight);
                __13.__.display();
                system("pause");
                  return 0;}
```

	Before taking the Programming course, I can solve the following problems.	**After taking the Programming course, I can solve the following problems.**
22. What is the missing code for blank number 10?	☐ Yes ☐ No	☐ **obj1** ☐ obj ☐ obj2 ☐ laundryobj
23. What is the missing code for blank number 11?	☐ Yes ☐ No	☐ Cost ☐ BasicCost ☐ Laundry ☐ **display**
24. What is the missing code for blank number 12?	☐ Yes ☐ No	☐ laundry ☐ laundr ☐ **Laundry** ☐ laund
25. What is the missing code for blank number 13?	☐ Yes ☐ No	☐ obj1 ☐ obj ☐ **obj2** ☐ laundryobj