STUDENT SUCCESS MODEL IN PROGRAMMING COURSE: A CASE STUDY IN UUM

SALAM ABDULABBAS GHANIM

MASTER OF DEGREE
UNIVERSITI UTARA MALAYSIA
2014

STUDENT SUCCESS MODEL IN PROGRAMMING COURSE: A CASE STUDY IN UUM

A thesis submitted to Dean of Awang Had Salleh Graduate School in
Partial Fulfillment of the Requirements for the Degree
Master of Science of Information Technology
University Utara Malaysia

By
Salam Abdulabbas Ghanim

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences

UUMCollege of Arts and Sciences

Universiti Utara Malaysia

Abstrak

Kesukaran dan kerumitan dalam pengaturcaraan komputer telah dianggap sebagai punca kadar kegagalan dan keciciran yang tinggi. Pengaturcaraan telah dianggap oleh pelajar novis dan pertengahan, malah pelajar cemerlang juga sebagai satu kursus yang memerlukan kaedah pembelajaran yang pelbagai dengan menghasilkan dapatan yang pelbagai. Faktor-faktor kejayaan kursus pengaturcaraan di institusi pengajian tinggi telah dikaji. Rekod di Universiti Utara Malaysia (UUM) menunjukkan 38% dari pelajar semester satu ijazah sarjanamuda yang mengambil kursus pengaturcaraan dalam tahun 2013 telah gagal. Ini merupakan motivasi bagi kajian ini, yang meletakkan matlamat untuk mengenalpasti faktor praktikal yang mempengaruhi kejayaan dalam kursus pengaturcaraan, dan untuk menokok dapatan teoritikal di kalangan dapatan-dapatan sediaada oleh kajian lain. Kaedah kuantitatif telah digunakan, dengan mendapatkan data dari 282 responden yang telah disampelkan di kalangan pelajar sarjanamuda dan sarjana Teknologi Maklumat (IT) dan Teknologi Komunikasi dan Maklumat (ICT). Setelah data ditapis dan dibersihkan, dengan empat rekod yang mengandungi data terpencil dihapuskan dari senarai, ujian-T bebas, korelasi, dan regresi dijalankan bagi menguji hipotesis yang telah dibentuk. Dapatan dari Korelasi Pearson menunjukkan alatan pengajaran, konsep OOP, motivasi, penilaian kursus, dan keupayaan matematika mempunyai hubungan positif dengan pencapaian akademik. Manakala, ketakutan mempunyai hubungan yang negatif. Analisis regresi seterusnya menunjukkan hubungan adalah kuat, kecuali hubungan negatif iaitu ketakutan dengan pencapaian akademik. Ujian-T bebas pula membuktikan perbezaan antara kumpulan yang telah mempunyai pengalaman dan yang belum mempunyai pengalaman tidak wujud.

Keywords: Pengaturcaraan berasaskan objek, Java, kesukaran pengaturcaraan, pembelajaran, faktor

Abstract

The complexity and difficulty ascribed to computer programming has been asserted to be the causes of its high rate of failure record and attrition. It is opined that programming either to novice, middle learner, and the self-branded geeks is always a course to be apprehensive of different studies with varying findings. Studies on factors leading to the success of programming course in higher institution have been carried out. The record at Universiti Utara Malaysia (UUM) shows that 38% of semester one undergraduate students failed the programming course in 2013. This really motivates this study, which aims at investigating the practical factors affecting the success of programming courses, and to position its' theoretically findings to complement the existing findings. Data were gathered using a quantitative approach, in which a set of questionnaire were distributed to 282 sampled respondents, who are undergraduate and postgraduate students of Information Technology (IT) and Information and Communication Technology (ICT). Having screened and cleaned the data, which led to the deletion of four outlier records, independent T-test, correlation, and regression were run to test the hypotheses. The results of Pearson correlation test reveal that teaching tools, OOP concepts, motivation, course evaluation, and mathematical aptitude are positively related to academic success in programming course, while fear is found to be negatively related. In addition, the regression analysis explains that all the elicited independent variables except fear are strongly related. Besides, the independent T-test also discovers no deference between groups with and without previous programming experience.

Keywords: Object Oriented Programming, Java, programming difficulties, learning, Factors

ACKNOWLEDGEMENT

In the name of Allah, the Most Gracious and the Most Merciful

Alhamdulillah, all praises to Allah for the strengths and His blessing in completing this thesis.

Special appreciation goes to my supervisor, Mdm Alawiyah Abd Wahab, for her supervision and constant support. Her invaluable help of constructive comments and suggestions throughout the success of this research. This thesis would not have been possible without her help, support and her patience.

I sincerely thank to my evaluators Dr. Mazni Omar and Ms. Rohaida Romli, for graciously reviewing this work and giving valuable suggestion and comments on my work.

My deepest gratitude goes Prof. Dr. Huda (Dean, College of Arts and Sciences), Dr. Norliza , Dr. Hizbullah and all administrative staff of school of information technology specially Madam latifah.

I would also like to say a big thanks all UUM lecturers and staff members at the School of Computing who were kind enough to give me their precious time and assistance, without which I would not have been able to complete this Master's Thesis.

I am indebted and thankful to the Chancellor of University Utara Malaysia who referred me to valuable e-resources at the Sultanah Bahiyah Library.

Sincere thanks to all my friends especially Nasser Jabir, was always willing to help and give his best suggestions. I have been a lonely without him, and others for their kindness and moral support during my study. Thanks for the friendship and memories.

Last but not the least, I would like to thank my family: my Mother. She was always there praying me. Also to my elder brothers, and elder sisters. They were always supporting me and encouraging me with their best wishes. Special thank to my cousin Mohammed Tuama, Be on the go to do any requirements in my country when i ask.

Finally, I would like to thank my wife for her personal support and great patience at all times. She was always there stood by me through the good times and bad.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE
INTRODUCTION

1.0 Background of the Study

Modern curriculum needs to emphasize the development of programming skills for citizens of a technological society (Pejcinovic, Holtzman, Chrzanowska, & Jeske, 2013). Programming is a cognitive activity that requires abstract representations and logical expressions. The program must translate abstract representations into correct codes by using a formal language to create, modify, reuse, or debug a program (Wiedenbeck, 2005). Furthermore, programming is often viewed as a problem-solving activity rather than a linguistic activity, often ignoring the fact that programming languages are a case of formal languages. The interpretation of formal languages is unique for every individual.

Programming skills are an essential part of computer science (CS) and information technology (IT) courses (Raina Mason, Cooper, & Raadt, 2012). Robins, Rountree, and Rountree (2003a) argue that programming skills are useful in programming knowledge and strategies, such as program generation and comprehension. Programming can also lead to a rewarding career, such as an analyzer, programmer, or debugger.

Zdancewic and Weirich (2013) state that programming is a conceptual foundation in the study of computations. Programming is a prerequisite for almost every other course in CS. Renumol, Jayaprakash, and Janakiram (2009) said that "*programming is the process of writing, testing and debugging of computer programs using different programming languages*." However, according to

Schreiner (2011), a program is the formal description of a method that solves a particular problem.

Programming languages have two basic levels: a high-level languages, which are classified into three groups, namely, procedural (C, C++, Visual Basic, and Java), non-procedural (LISP and PROLOG), and, problem oriented (MATLAB, MATHEMATIC, and LATEX); a low-level languages, such as machine language and assembly language (VRajaraman, 1998). Matravers (2011) argues that the low-level representation of a central processing unit instruction set is known as the machine language of a computer. Thus, directly writing instructions in binary form is difficult.

A programming language is the usual way of presenting a paradigm to allow the programmer to write a program that solves a certain problem (Rinard, Scales, & Lam, 1993). A programming language is a formal representation of a program. A program may be written in different programming languages, similar to a human thought that may be formulated in different human languages (Grogono, 1989). According to Li, Liu, Mao, and Zhou (2013), the program derivation process begins with an informal specification of a given problem. Thereafter, the informal specification is formalized in terms of pre-conditions and post-conditions.

Teaching programming at the university level has been the basis for many lively discussions among CS teachers (Moderator, Koffman, Kölling, & Reges, 2005;Bailie, 2003;Bruce, 2005). Furthermore, it is not an easy task (Renumol *et al.*, 2009). Students typically encounter early challenges when learning programming for the first time. These difficulties arise because programming is mainly taught by using an intuitive approach that treats programming more as an art than a science. Novices learn programming in a “trial and error” or “guess and test” manner. Thus,

novices obtain little confidence on program development and obtain a “fear” of practicing programming (Li, Liu, Mao, & Zhou, 2013). An individual requires procedural knowledge in computer programming to write a program. Renumol (2009) says that knowledge of programming language semantics and syntax, which requires comprehension and memorization, is necessary. In addition, he stressed that program design and problem solving skills , which require extra skills such as domain knowledge, logic, and abstraction, are also needed to be programmer. Therefore, programming is a difficult undertaking that requires several computer skills and knowledge. Studies on programming education argue that the dropout and failure rates of programming courses are comparatively high (Bergin & Reilly, 2005; Bennedsen & Caspersen, 2007) and that their overall effectiveness is poor.

Tutors spend a significant amount of class time explaining fundamental computer language concepts and relevant algorithms to computer programming students (Carlisle, 2009). However, certain novice students learn their first computer language without any difficulty, whereas others struggle and require considerable support and assistance from tutors (Garner, Haden, & Robins, 2005). According to Robins, Rountree, and Rountree (2003b), these differences between novices can be attributed to their past knowledge, strategies, and mental models of the programs. This mean there are factors and skills that effect on students abilities in their programming learning.

The object-oriented programming (OOP) paradigm has been taught in different university departments either as an introductory programming course or a subsequent programming course in the last few years (Sivasakthi & Rajendran, 2011a; Xinogalos, 2006).

However, most studies on OOP education show that students often confuse elements in a programming language such as: object, class, attributes and methods. Furthermore, students face difficulties in implementing solutions to specific problems by using other programming language (Holland *et al.,* 1997).

This study was focuses on Java as OOP. According to Bennett, Fisher, and Lees (2011), no differences exist between OOP and the restructuring of a high-level world view where the object in OOP has attributes are same the attributes of the object in the real world (i.e car has name, color, model, etc). Furthermore, Poo and Ashok (2007) state that OOP set data and operations into units called objects and allowed objects to be combined into systemic networks to build a program. Objects and their interactions are the main elements of program design in OOP. Each object has a state (data) and a behavior (operations on data).Thus, Objects in OOP are not much different from ordinary physical objects.

OOP is a method of software enhancement wherein the form of the program depends on objects and on objects interacting with each other to achieve a task (Sajaniemi & Kuittinen, 2003). Java programming language is very well established (Madden & Chambers 2002). Consequently, Sivasakthi and Rajendran (2011) state that OOP, particularly Java, has become taught to undergraduate and postgraduate IT, ICT, and CSE students.

Factors that influence programming education have been identified over the years (Wiedenbeck & Labelle, 2004). As well as they add the following: "*we still far from a full understanding of why some students learn to program easily and quickly while others flounder.*" Factors such as cognitive engagement, learning process, computing tasks (Carbone & Hurst, 2009), spatial ability, mathematical aptitude (Patil, 2009), knowledge, aspirations, dispositions, perceptions, expectations, skills,

values, needs, and goals (Helme & Clarke, 2001) contribute to the propensity of the student to learn. Student success model has been the finding of this study contained five factors which are teaching tools, motivation, course evaluation, OOP concepts and student aptitude in mathematics. Survey made up from 282 post/undergraduate students in UUM enrolled in IT and ICT departments. The model seeks to help instructors to improve their approach in teaching programming course, as well as policy decisions makers by consideration the mentioned factors, where they have affect the academic success of students in programming course. The model was based on the students' perceptions (more details in section 5.1).

1.1 Problem Statement

Computer programming is an area that is both complex and difficult (Rainalee Mason, 2012). According to El-Zakhem and Melki (2013) and Rainalee Mason (2012), most CS students face major problems in their first programming course. Furthermore, Dehnadi and Bornat (2006) state that programming is difficult to learn. Educating novices on programming has been considered a big challenge since the early 1970s (Floyd & London, 1970;Gries, 1974; McCracken *et al.,* 2001; Robins *et al.,* 2003b; Spohre 1989;Wenger, 1998).Teaching programming is considered one of the seven grand challenges in computing education (McGettrick *et al.,* 2005).

First year students encounter a wide variety of challenges in learning objected oriented programming, including understand the principles of OOP such as (Data Abstraction, Polymorphism, Encapsulation and Inheritance) and the efficient design of programs (Butler & Morgan, 2007). As well as, Sharp and Schultz (2013) find that learning OOP is difficult for students because it requires skills of comprehension and memorization abilities; the latter involves high-level abilities, which require additional skills such as abstraction, encapsulation, polymorphism, and inheritance. In addition, Biju (2013, p. 1) state that *"Understanding object oriented concepts is always a difficult task for students. It is equally challenging for lecturers to teach these concepts".*

Sivasakthi and Rajendran (2011) observe that students have learning difficulties on Java programming topics. For example, Milne and Rowe (2002) state that students will struggle in learning programming until they obtain a clear mental model of how programming "works," that is, how programming is stored in memory and how the objects in memory relate to each other.

An international survey of introductory programming teachers conducted in 2006 found that Java was used as the first language by 58% of respondents followed by C++ at 18% and Pascal at 9% (Schulte & Bennedsen, 2006). It can be clearly show that java widely usage comparison with the other programming languages.

Butler and Morgan (2007) indicated that introductory computer programming has been studied extensively in a wide range of technical and educational facets. Numerous studies have also focused on OOP programming. However, these studies do not focus on the challenges faced by first year students with Java as OOP. Furthermore, Eckerdal (2006) mentions that the problems encountered by students include the increasing complexity of programming languages such as Java.

According to the aforementioned statements and statistics provided by the ASIS (*Academic and Student Information System*) at the final semester of 2013, found that 38% of students who took programming courses in Universiti Utara Malaysia (UUM) obtained a grade of C- or below. Thus, the classification of this rates of student failure are considerable (Butler & Morgan, 2007). Therefore, this study attempts to identify and investigate the significant factors that affect the propensity of UUM students to learn Java programming as OOP. Furthermore, this study addresses the lack of information from previous studies.

1.2 Research Questions

1. What are the factors that may affect the academic success of students in computer programming?
2. How to develop student success model based on the factors that have affected the academic success of students in computer programming?
3. How to evaluate the student success model?

1.3 Research Objectives

1. To identify the factors that affects the academic success of students in computer programming.
2. To develop student success model based on the factors that have affected the academic success of students in computer programming.
3. To evaluate the student success model.

1.4 Significance of the Study

The researcher expects that this study will be significant in several areas. First, this study will add to literature on the academic success of students in computer programming by identifying factors that may affect the academic success of students in computer programming. This research helps academics who are interested in understanding the factors that affect the propensity of students to learn programming because only a few similar studies have addressed this particular issue.

Additionally, based on understanding of significant factors that affect the academic success of students undertaking programming course in UUM, this study attempts to provide recommendations to programming instructors on how to improve their approach to teaching programming. The researcher anticipates that by implementing the recommendations failure rate in the OOP subject particularly in Java programming could be reduced.

1.5 Scope of the Study

Although OOP languages exist, this study has been focused on the Java OOP programming language (see Section 1.1). In addition, Java programming is popular both in Academia and the IT Industry. Further, it is the most used programming language across the world (Bennedsen & Paterson, 2007; Sivasakthi & Rajendran, 2011).

Nikishkov, Nikishkov, and Savchenko (2003) argued that Java is completely OOP. On the other side, Singer, Li, and David (2013) stated that Java is the most used programming language in educational institutions. According to many researchers, The time, cost and willingness of the participants important criteria for researcher to identify the scope (Sekaran, 2003; Creswell, 2009). Therefore, this

study highlighted on the factors that affect academic success in computer programming. Thereby, the effect of these factors on UUM post/undergraduate students (IT/ICT) taking programming courses has been investigated. This study was broadly included students in UUM.

1.6 Organization of the Research

This study is organized into five major chapters:

Chapter One: Introduction. The first chapter constitutes the background of the study and highlights the definitions of programming, program, and programming languages. This chapter also presents some of the difficulties and challenges that students encounter when studying programming. Furthermore, the research objectives and research questions are covered and the rationale of the study is explained.

Chapter Two: Literature Review. The second chapter comprises the literature review of this study. All studies on programming and studies that emphasize significant programming factors, such as teaching tools, experience with other programming languages, OOP concepts, motivation, course evaluation, fear and mathematical aptitude are reviewed. These factors have been considered in this study for investigating them whether they effect on academic success in computer programming.

Chapter Three: Research Methodology. The third chapter explained the methodology used in collecting relevant data for this research. This chapter was also explained the sampling procedure and statistical approach used to analyze the data.

Chapter Four: Data Analysis and Research Findings. The fourth chapter addresses the data analysis stage of this study. It presents the data analysis process as done stage by stage in view of answering the earlier elicited research questions.

Chapter Five: Discussion, Conclusion and Recommendation. The fifth chapter concludes this study. It entails the discussion of the research findings, interpretation of the entire result of this study and the accompanying discussion. It argues the position of the findings of this study amidst the previous studies' findings.

CHAPTER TWO
LITERATURE REVIEW

2.0 Introduction

This chapter covers the literature review, including the OOP definition, related studies in this field, and factors considered to the objectives of this study. Literature reviews play important roles in shaping the research problem because the literature review process helps researchers understand the subject area, thus helping to conceptualize the research problem clearly and precisely. The literature review also makes the research problem relevant and pertinent to the field of inquiry (Smith, 2012). Also, he suggested that the researcher should start with general information and gradually narrow it down to the specific.

2.1 Object Oriented Programming

Programming knowledge includes skills and concepts such as problem investigation, problem-solving design, transformation of the design into code and data structure by writing a highly constrained language, and verification of the validity of the program (Herman & Salam, 2011). In recent years, OOP has become the most influential programming paradigm. OOP is widely used in education and different industries; furthermore, almost every university includes object orientation in the curriculum (Sivasakthi & Rajendran, 2011b). Learning to program is notoriously difficult. For instance, Bergin and Reilly (2005, p.293) note that "*it is well known in the computer science education (CSE) community that students have difficulty with programming courses and this can result in high dropout and failure rates.*" . At the same time, according to many researchers, teaching programming to

novices has been considered a big challenge for almost 40 years (Floyd & London, 1970;Gries, 1974; McCracken *et al.*, 2001; Robins *et al.*, 2003b; Spohre 1989;Wenger, 1998). Teaching programming is considered one of the seven grand challenges in computing education (Mcgettrick *et al.*, 2005). According to Diederich (1988), the relevant elements in teaching can often be described by the didactic triangle (Figure 2.1). Unfortunately, more studies concentrate on the teacher as the substantial factor, and few studies focus on the students and content. Many published research materials on the Java programming language mostly focus on technology issues and related enhancements. Therefore, this study seeks to fill this gap in literature by identifying factors that affect the teaching of programming from the perspective of students.

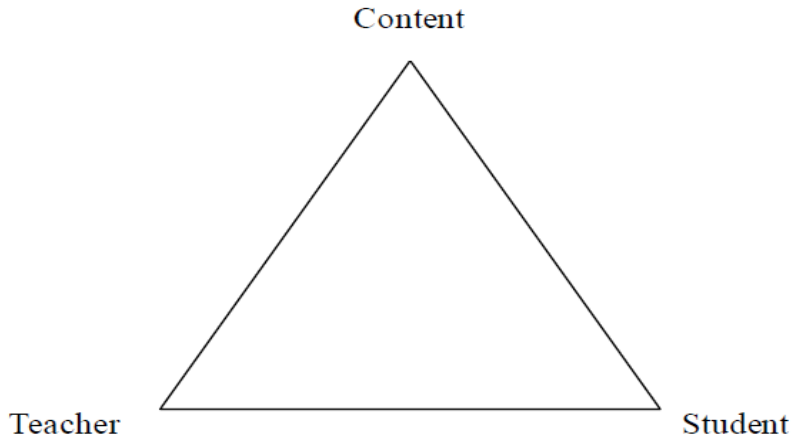

Figure 2. 1 The didactic triangle (Diederich, 1988).

Madsen and Møller-Pedersen (1988, p.16) defined OOP as follows: *"a program execution is regarded as a physical model, simulating the behavior of either a real or imaginary part of the world."* OOP is a clever concept and has become a very common term (Henderson & Zorn, 1994). As stated beforehand, OOP was used as the first language in most universities, in particular Java programming

(Schulte & Bennedsen, 2006). However, numerous studies focus on how to develop OOP programming learning.

According to El-Zakhem and Melki (2013) and Rainalee Mason (2012), most CS students face major problems, such as in OOP principles (i.e object, class, attributes and methods) and efficient program design (Butler & Morgan, 2007),during their first programming course.

Some researchers have published articles that describe the factors affecting introductory programming students. However, studies on OOP that takes students as the main sample are lacking. According to the annual statistic conducted in UUM in 2013, 38% of students undertaking programming courses got C− or below; thus, the student failure rates are considerably high (Section 1.1). This study strives to identify and investigate the significant factors that affect the propensity of UUM first year students to learn Java programming as OOP.

Georgatos (2002, p.3) noted that programming "*is a human activity that is a great challenge, involving the design of machine behavior that can assist, and at times replace, humans in tasks of intellectual nature*."

The product of this activity is a "program" that can be different things at different times:

- The program can describe calculations; the *imperative* or *procedural* programming model.
- The program can describe and treat objects; the OO programming model.
- The program can define functions; the *functional* programming model.
- The program can define logical relationships; the *logical* programming model.

Another definition of a program is syntactical which means, a program is a text constructed according to certain grammar rules (Pair, 1993). Programs are always full of errors and debugging takes time because tracking bugs and correcting them is often difficult. OOP is a programming technique and a paradigm for writing "good" programs for a set of problems. Only some programming languages are "OO" (Stroustrup, 1991).

2.2 Java Programming

Sivasakthi and Rajendran (2011b, p.1) state the following: *"Java programming is popular both in Academia and IT Industry. Further, it is the maximum usage of programming across the world."* Moreover, given the new possibilities provided by Java for the web, the Java paradigm has received considerable attention. Thus, many universities and colleges have introduced Java into their undergraduate and postgraduate CS curriculum (Said Hadjerrouit, 1998). Thus, the teaching and learning Java programming in academia has become a great responsibility. Madden and Chambers (2002) adds that the Java programming language is very well established and is often the first object-oriented (OO) language taught to students.

Despite the popularity of programming languages such as Java, issues still exists on the suitability of these languages for education, particularly in the introduction of programming to novices (for instance, Mody, 1991; Said Hadjerrouit, 1998;Biddle & Tempero, 1998;Close, Kopec, 2000;Clark, MacNish, & Royle, 1998). Pears *et al.*, (2007) state that Java is not designed for educational purposes compared with Python, Logo, Eiffel, and Pascal.

This study aims to investigate the factors that cause the learning difficulties of students with regard to Java as OOP. Java has become the most influential programming paradigm in recent years. Although empirical studies of programmers and programmer comprehension have been conducted with regard to procedural and OO languages, few studies have been conducted to discover the individual traits cause the most difficulty to novice programming students (Milne & Rowe, 2002).

Java programmers generally require declarative and OO knowledge (Sivasakthi & Rajendran, 2011a). The former involves knowledge on Java programming language syntax and semantics, which require comprehension and memorization abilities; the latter involves high-level abilities, which require additional skills such as abstraction, encapsulation, polymorphism, and inheritance. Consequently, learning OOP by using Java includes many challenges (Butler & Morgan, 2007) and requires multiple skills and types of knowledge. This study used data collection and analysis to identify the various learning difficulties involved in Java programming. Many researchers have argued against the use of Java. These researchers highlight the inherent difficulties in using Java as a first programming language (Hadjerrouit, 1998; Crawford & Boese, 2006; Powers *et. al.* 2006;Gross & Powers, 2005).

Teaching Java is challenging (Nedzad & Yasmeen, 2001). Gosling (1996) noted that Java is a general-purpose OO language that is designed to be simple to enable many programmers to achieve fluency in the language. Iain and Glenn (2002) state that students will struggle to understand this language until they gain a clear mental model of how programming "works," that is, how programming is stored in memory and how objects in memory relate to one another.

Certain interactive and integrated environments such as “BlueJ,”“Greenfoot,” and “Processing,” make Java programming easy to learn and teach. The following section will discuss these tools in detail. A survey of Java textbooks supported by a survey of student perceptions regarding the difficulty of various topics has yielded a hierarchy of topics from the least difficult to the most difficult: comments, output, assignment, expressions, if- statements, for-loops, arrays, methods, classes, and input (Yau and Joy,2004).

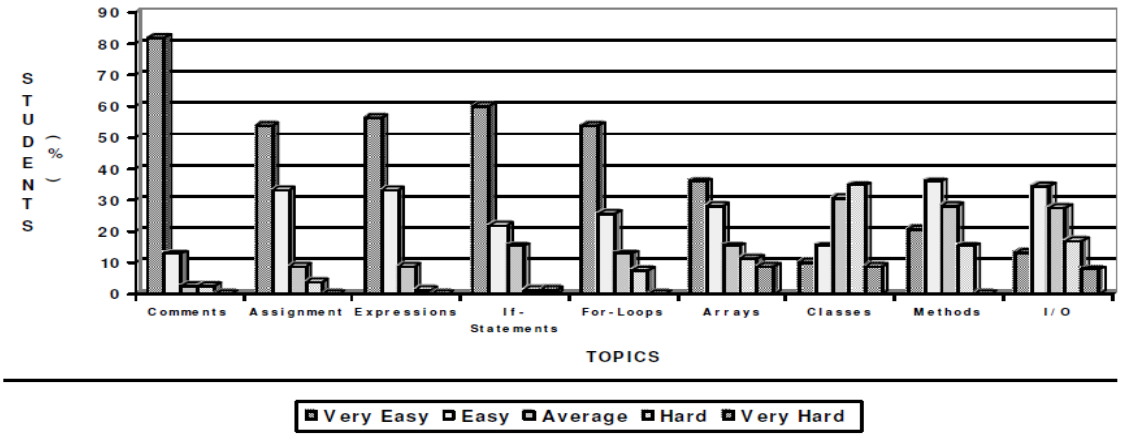

Figure 2. 2 Level of learning difficulties on different topics of Java programming.

However, Herman and Salam (2011) stated that novice students often face difficulties in learning programming because of various issues and the nature of the subject, which can be vague and invisible. This research focuses on UUM students, most of which are programming novices. Mow (2008) refers to the differences between novices and experts in the following:

- Novices have difficulty recognizing incorrect grammar and struggle with syntactic knowledge, whereas experts readily recognize grammatical errors.
- In terms of semantic knowledge, experts have effective mental models of virtual or notional machines, whereas novices have yet to build these models.

- For schematic knowledge (knowledge of the structure of a program) experts use deep structures to categorize programs based on the type of routines required. By contrast, novices use superficial features for categorization.

Furthermore, novices are inclined to use low-level plans and are unskilled at problem decomposition, whereas experts maintain an overall view of the problem in mind while decomposing problems into small, manageable sub-problems. Experts also consider more alternative solutions and are more adept at comparing different solutions than novices (Lahtinen, Ala-Mutka, & Järvinen, 2005).

2.3 Related Works

The high dropout and failure rates in programming subjects have drawn the attention of researchers. A number of papers have also been written to address problems that occur when teaching Java. However, most of these studies focused more on the teacher than on the student. Therefore, this section elaborates on studies that are related to this field. A debate is taking place in many computer/information science departments on the best approach to teach programming. Students should be exposed immediately to the new OOP paradigm by using a language such as Java (Burton & Bruhn, 2003). Therefore, this section will also discuss the methods used in previous studies to solve difficulties in programming course.

There are many studies achieved in teaching the programming languages, such conducted by Byrne, Catrambone, and Stasko (1999) who use two experiments designed to test whether animating algorithms will assist students to learn algorithms effectively. However, this study focused only on software visualization (teaching tools). While, Wilson and Shrock (2001) state that many factors affect the success or failure of students in programming.

The study conducted by Byrne and Lyons (2001) focuses on the BASIC programming language and their data has been gathered from academic records. In contrast, this research concentrates on the OOP as noted by Wiedenbeck (1999) the choice of programming language affects the understanding of programming.

Furthermore, Milne and Rowe (2002) have investigated C++ and asked both tutors and students on the individual concepts of the programming language they strive to teach and learn, the conclusion of their study is the motivation to design a program visualization tool. As well as, they focused on participants who have experience in programming languages, while, this study deals with the novice students.

Wiedenbeck and Labelle (2004) investigated the combined effects of mental model, self-efficacy, and prior experience on programming learning. However, the respondents came from different disciplines. Most respondents are not involved in CS.

Wiedenbeck (2005) concentrated on important factors that affect program learning: perceived self-efficacy, knowledge organization, and prior programming experience. The differentiation of Wiedenbeck study was non-major students were his participants. Programming courses are complex for a majority of university students, particularly students who have little previous exposure to programming.

In addition, Bennedsen and Caspersen (2006) focused on learning OOP. Their study depended on the perspectives of lecturers and that of the university administration. By contrast, this research concentrated particularly on the perspective of students. On the other hand, study by Caspersen and Kölling (2006) aimed to assist novice programmers learn better and faster. in the same time, laying the foundation for a thorough treatment of the aspects of software engineering. Their

study does not identify the factors that affect how students learn programming and instead focuses on the programming process (Concepts of the program).

Carbone and Hurst (2009) discussed the internal domain factors which are motivation and capability that influences how students learn programming. In their study, data was gathered from few students by semi-structured interviews. The results of their study depend on how the students deal with an introductory programming language. By contrast, This research strives to identify the significant factors that affect the success of Java as the OOP.

Li, Liu, Mao and Zhou (2013) concentrated on the factor of fear. However, their study focuses on non-CS major students and on the experiences of researchers as lecturers. Thus, this study was focused directly on IT and ICT students.

Additionally, studies into whether choice of programming language affects program comprehension are well documented, and have shown that different notations facilitate the understanding of different kinds of information found in programs (Wiedenbeck, 1999). Other studies have conducted research into the types of mental models formed by both novice and expert programmers, and how such models affect their understanding of the problem and its solution (Burkhardt *et al*., 1997;Blackwell, 1996; Turner, 2001). Unfortunately, few researchers have examined the OOP learning experience of programming students and the difficulties that they face in their field.

Research on success factors has been conducted in the sub-areas of introductory programming and in general CS education. Studies have identified the problems and solutions in programming education. However, these studies have mostly focused on programming education in foreign countries such as Australia, Finland, the United States, and the United Kingdom. Therefore, this study has been

conducted in Malaysia, particularly in UUM. The perception of students toward their learning problems and their ideal OOP learning method was examined in this study.

2.4 Student Success Model in programming Course and Hypothesis

Programming students are strained by the learning challenges in their field (Astrachan, Selby & Unger 2006;Garner, 2001). Several studies have tried to modify the OOP teaching and learning mechanisms, particularly for Java, to help students overcome such challenges. Most novice programmers still struggle to become proficient in the subject (Mow, 2008). Therefore, this study was focused on the following factors and investigate them whether they have affected the academic success of novice students in computer programming. Eventually, use them to develop student success model (Figure 2.3):

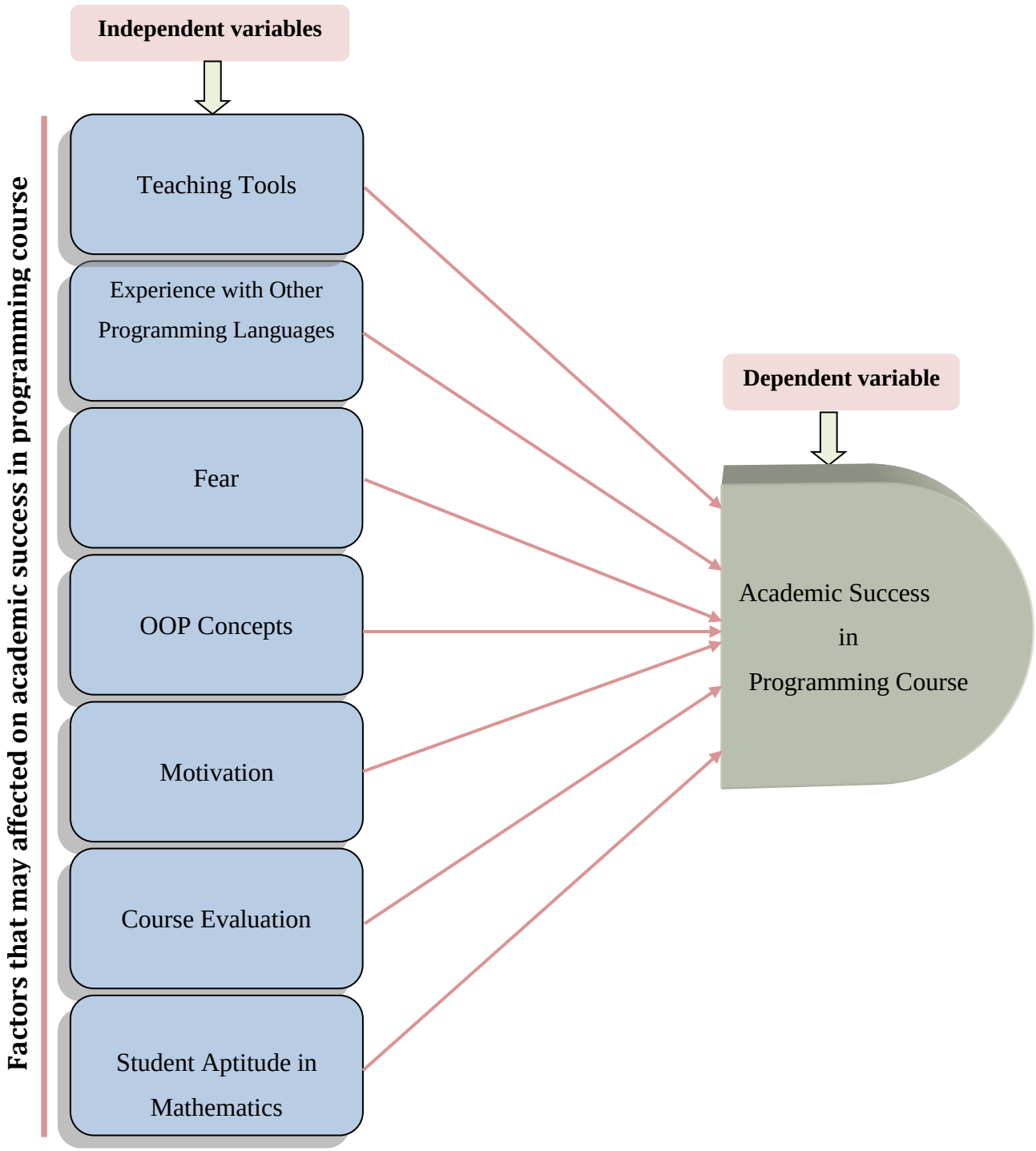

Figure 2. 3 Factors that may affect the academic success in programming course.

2.4.1 Teaching Tools

Teaching programming languages is a challenging issue with a long history (Costa, Aparicio, & Pierce, 2009;Ulloa, 1980). The teaching of OOP is undertaken using a combination of lectures, some tutorials and supervised labs (Madden & Chambers, 2002). Graf, Lan, and Liu (2009) found that programming students adopted different ways to learn their subject. Some programming students may regard individual learning as the most suitable learning process. However, Jenkins (2002) found that some students opted for a dynamic learning environment (Classroom), which greatly improved their learning by studying with their peers. Moreover, programming instructors are having difficulties in instilling favorable programming habits into their students. Ulloa (1980) found that interactive software can be automated to help the instructors in teaching their students individually and in solving their problems regarding the subject.

Many researchers have suggested the use of functionally reduced development environments (e.g., DrJava(Reis, 2004) and BlueJ (Kölling, 2003)) that are specifically designed for educational purposes. These tools assist programming students by providing them with clear descriptions of Java programming mechanics. Some researchers have proposed the use of environments that support the visualizations and animations of computational elements that are based on structures and simple command syntaxes, such as Karel-3D (Brusilovsky, Calabrese, Hvorecky, Kouchnirenko, & Miller, 1997) and Alice3D (Cooper, Dann, & Pausch, 2003).

Some researchers have suggested the adoption of visualization and interaction techniques for the creation of interactive environments that provide

freedom to programming students to explore their field. Milne and Rowe (2002) argues that a program visualization tool can help programmers interpret the processes occurring in memory while a specific program is being operated. The adoption of teaching tools significantly affects the understanding of programming students of their subject. Several studies have identified numerous problems that are associated with OOP instruction and that concerns different aspects of the adopted systems (Black *et al*., 2013;Kölling, 1999).

2.4.2 Experience with Other Programming Languages

Students who are about to enter IT-related university courses are expected to have basic computer literacy (Wit, Heerwegh, & Verhoeven, 2012) and programming experience (Hardy, Heeler, & Brooks, 2006). Study by Howles (2007) have examined the number of students who have no programming experience and have limited computer usage take an IT-related course in the university. (Thompson, 1995) argues that the previous experiences of an individual can significantly affect his or her collection and use of knowledge. Therefore, this theory defines learning as an active, subjective process that allows students to create knowledge from their previous experiences or by extending their present knowledge.

According to Armoni, Gordon and Harel (2012), only few studies have examined how the previous experiences of students affect their understanding of programming languages and models. Several academic programs have been developed to suit all IT students despite their varying knowledge of other programming languages (Madden & Chambers, 2002). However, not all of these programs focus on OOP. Yau and Joy (2004) argues that Java should not be

taught to students with no programming background given that OOP paradigms are highly abstract than procedural paradigms. Armoni *et al.*, (2012) also argued that the programming background of a student could influence his or her attitude toward learning another programming language.

2.4.3 Fear

The teaching and learning processes in the CS and IS fields have received significant research attention because of the high attrition rates of these courses (Robins *et al.*, 2003a). However, the educational environment of these courses remains a global problem in the field of computer programming (Mead *et al*., 2006).

Study by Rogerson and Scott (2010a) defines fear as the lack of appreciation or interest of students toward the programming subject. This term may also refer to the apprehension or the lack of confidence of these students on their programming knowledge. Programming is defined in this study as the full cycle of systems development, including the coding process, the use of basic theoretical concepts, and preparation of the final product for implementation (Bruce *et al*., 2004). Rogerson and Scott (2010, p. 148) defined this fear as *"experiencing a lack of confidence or apprehension regarding their ability to code or program"*. Several researchers have used the same term to describe the anxiety that some students feel when developing a program and to describe their feelings of discomfort that may reduce their interest in the subject (Simon *et al*., 2006). Bergin and Reilly (2005b) stated that such feeling of discomfort will discourage programming related inquiries and discussions from these students.

2.4.4 OOP Concepts

Students are greatly challenged by several elements in the Java programming language. The inclusion of Java in programming courses has been the subject of several studies and experience reports over the previous decade (Sivasakthi & Rajendran, 2011a; Xinogalos et al., 2006; Madden and Chambers 2002). These reports suggest that the use of diagrammatic representations can help students improve their understanding of OOP, such as UML or other analogous notations (Object Management Group, 2003; Alphonce & Ventura, 2002). Sicilia (2006) argued that programming instructors should carefully help their students in comprehending the OO concepts and in translating the conceptual models into Java programs. The OOP learning of these students is also hindered by several factors, such as their associations, generic containers, and differences between interfaces and classes. Moreover, Madden and Chambers (2002) asserted that, comprehension of a list of broad Java language tools such as (e.g. syntax, file handling, inheritance, Appletviewer, JCreator, GUI programming, etc) help to understanding OOP concepts. Many researchers have explained the OOP concepts as shown below:

I. **Object**

Object refers to the main component of the OO paradigm that is used for carrying out specific tasks (Garrido, 2003). Actual examples of an object include a bus, a book, or a student. Therefore, people think about, identify, act upon, or assign concepts to several objects on a daily basis (Satzinger & Ørvik 2001).

II. **Class**

Students must clearly differentiate the concept of "object" from the concept of "class" (Eckerdal, Box, & Thun, 2005). The latter refers to a general category, whereas the former refers to a specific instance (Satzinger & Ørvik, 2001). Objects are grouped together into classes that specify the type of an object, whereas a class can be used as a template for a potential object (Weisfeld, 2004).

III. **Attributes and Methods**

Both attributes and operations are equally important in the OO approach. The former refers to the descriptive properties of an object that represent its state, whereas the latter refer to an operation determines the behavior of an object or what the object can do (Havenga, 2006).

IV. **Constructors and Destructors**

Constructors and destructors are special methods that play important roles in OOP. Constructors are used when creating new objects for the allocation of memory and the initialization of variables. Sebesta (2004) referred that, Delphi uses the "create" constructor to create an object. Additionally, he state that, All objects in Java are explicit heap dynamic (i.e., created explicitly on the heap during runtime) and are allocated into the new operator. Destructors are used to reclaim the heap storage and to destroy objects. Instead of using a destructor, Java uses an implicit garbage collection process that does not require the programmer to create a code for the destructor (Havenga, 2008).

V. **Abstraction and Associations**

Abstraction refers to the ability of an individual to define and use variables and operations that ignore several details. Abstraction aims to simplify the presentation of entities and to reduce their complexity during the programming process (Sebesta, 2004). Abstraction is classified into process abstraction and data abstraction. The former refers to the calling of a subprogram (method/procedure/function) without providing its details, whereas the latter refers to the declaration of the type and the operations in objects that are contained in a single unit, which restricts data access by sending messages to the methods (Schach, 2005;Sebesta, 2004).

VI. **Polymorphism and Dynamic Binding**

Polymorphism refers to the provision of multiple forms and methods. When used in the OOP context, this term implies that different objects may respond individually to the same message. Therefore, polymorphism may be used to indicate different implementations (Weiss, 2000). Polymorphism also supports greater abstraction wherein a single message can evoke different behavior (Rosson & Alpert, 1990).

2.4.5 Motivation

Helme and Clarke (2009) stated that students need motivation (the will to learn) and skills (capability) in order to be successful in their respective fields. Williams (2011) argued that the learning methods, motivation, and expectation of students can significantly effect to their learning. Several studies have identified the motivation and attitude of these students to learning as the core influential factors to

their successful learning (Gomes & Mendes, 2007;Jenkins, 2002;Robins *et al*., 2003b;Simon *et al*., 2006). Moreover, Jenkins (2001); Bergin and Reilly (2005) pointed that, the motivation can encourage the students to learning programming language well. In this case, motivation can be divided into intrinsic, extrinsic, and achievement motivation (Entwistle, 1998).

- Intrinsic motivation is present when the individual is interested and curious about the activity that he or she is currently performing;
- Extrinsic motivation is present when the individual anticipates a reward after successfully completing the activity; and
- Achievement motivation is observed when the performance of an individual is better than that of his or her peers.

Carbone, Hurst, Mitchell, and Gunstone (2009) found that intrinsically motivated students generally display higher programming capabilities, whereas externally motivated (i.e., passing the course) or achievement-motivated (i.e., obtaining higher marks) students do not cognitively engage themselves into the subject.

2.4.6 Course Evaluation

A non-personalized learning environment (Gomes & Mendes, 2007b) poses additional learning-related problems to students, reduces their motivation, and weakens their cognitive abilities (Simon *et al.*, 2006). Students rarely receive feedback or explanations from their instructors given their lack of time and the large class sizes in universities. The failure of instructors to pay individual attention to their students and to address their learning styles poses additional problems (Jenkins, 2002). Souza *et al.*, (2008, p.75) observed that the struggle

among programming students in their learning *"affects most facets of their study, for example: their progress through their study program, their study habits, their confidence and their time management."* Programming students must not merely rely on their textbooks to develop programs successfully (Gomes & Mendes, 2007; Jenkins, 2002; Lahtinen *et al.*, 2005).

Instructors should consider using programming patterns and playing games with their students to help improve their problem-solving abilities (Wangenheim & Shull, 2009). In addition, Madden and Chambers (2002) suggested that, it is important to ask students whether they found the course useful and enjoyable. Also whether they believe the course is well tied between theory and laboratory work.

2.4.7 Student Aptitude in Mathematic

Aptitude are usually used to refer to behavior that is used to predict performance or future learning (Macklem & Gayle, 1990). Students are required to study mathematics materials throughout their studies. Mathematic is used in multiple subjects such as finance, physics and computing, so attaining higher knowledge depends on the student's background in mathematics primitive. In addition, IT is a discipline that needs to build alliances with other disciplines, Mathematics an obvious alliance for IT, consequently, IT might learn from mathematic by developing options in computing (Guthrie, Yakura, & Soe, 2011).

Furthermore, Patil (2009) state that students admitted with passing level in mathematics aptitude show significant effects in development factors as well as programming skills. Also, he argues there is considerable enhancement in spatial

ability and hence programming ability of student groups, having mathematics aptitude up to passing level.

On the other hand, There are significantly need the mathematical skills in programming learning where, programming is based on new mathematical foundations which identifies the programming process with a step by step expansion of mathematical functions into structures of logical connectives and sub functions (Mills, 1972). As well as, Cheney and Kincaid (2012) state that aim of mathematics aptitude is to examine the underlying algorithmic techniques so that students learn how the software found the answer. Quenemoen,Thompson and Thurlow (2003) State that, in mathematics, decisions are made about how many items test basic students' programming skills and how many items test their problem-solving abilities. While, Jenkins (2002) argued that students who find programming difficult are simply and solely those for whom programming is difficult. There is nothing inherently difficult in the subject; it is simply that some students have no aptitude in mathematics. The skills often cited are problem solving and mathematical ability. Similarity Byrne and Lyons (2001) that hint link between mathematics ability and programming is widely accepted. In addition, Jenkins (2002) add that it is important to give students some exercises that involve simple mathematical manipulation such as: stock levels, collections of student marks, bank account details or baseball statistics.

2.5 Summary

This chapter presented a review of literature focused on the objected oriented programming, Java programming, related works in which how previous studies dealt with programming and what the attempts that used to decrease student challenges in programming learning. Finally, the student success model and hypothesis of the factors that may affect the academic success of student in programming course which are: teaching tools, experience with other programming languages, fear, OOP concepts, motivation, course evaluation and student aptitude in mathematics.

CHAPTER THREE
RESEARCH METHODOLOGY

3.0 Introduction

This chapter presents and justifies the methodology and design of this research. This chapter also presents the hypothesis, research method, data collection, population and study sample, research instrument, pilot test, data analysis and validity and reliability of the instrument development (questionnaires).

The strategies for answering the research questions must be identified after planning the research design (Smith, 2012). Figure 3.1 shows the strategies adopted in this study:

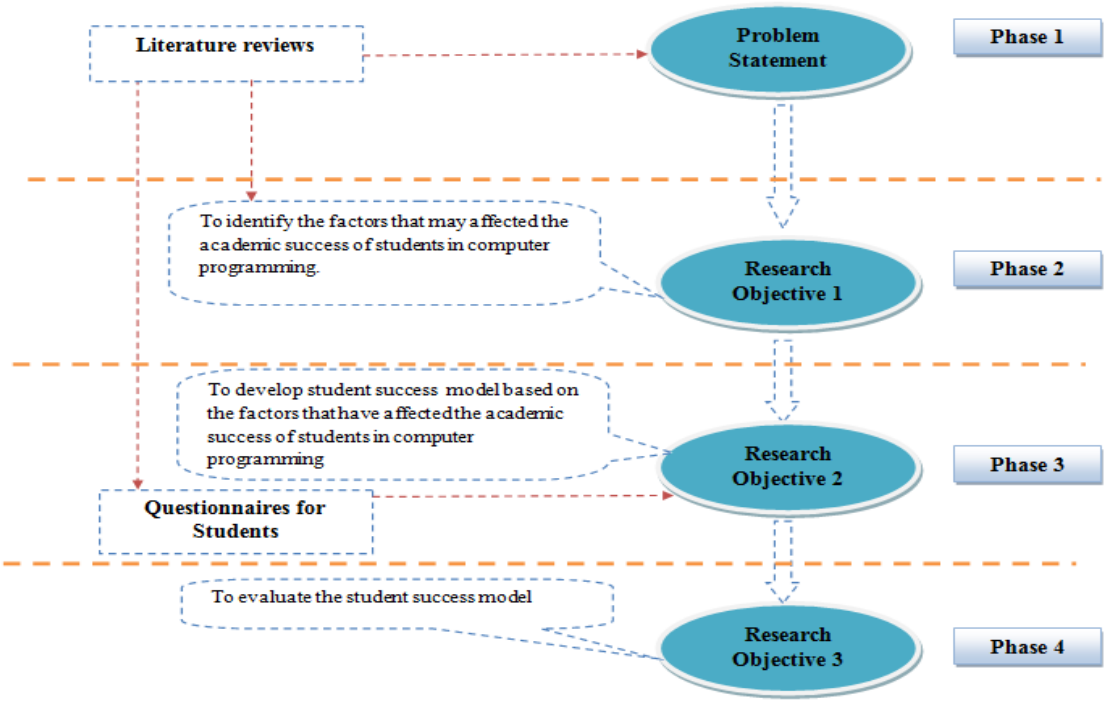

Figure 3. 1 The strategies that are adopted in this study.

3.1 Hypothesis of the Study

H1: *Teaching tools can affect the academic success in programming course.*

H2: *Experience with other programming languages can affect the academic success of programming students.*

H3: *Fear can affect the academic success of programming students.*

H4: *OOP concepts can affect the academic success of programming students.*

H5: *Motivation can affect the academic success of programming students.*

H6: *Course evaluation (which include: lectures, laboratory work, tutorials, and assignments) can affect the academic success of programming students.*

H7: *Mathematical aptitude can affect the academic success of programming students.*

3.2 Research Method

Quantitative data are expressed in numerical and statistical figures, which are analyzed and measured through statistical analyses (Hossein, 2007). The quantitative research design is used in this study to examine the responses from a large sample with regard to the proposed phenomenon. This research design also allows the researcher to analyze the behavior of respondents (Lakshman *et al.*, 2000). Questionnaires are used as the main data gathering tool for this research.

Smith (2012) stated that quantitative research design can validate the conclusion of the study by verifying the established concept and by proving or disproving a proposed concept. Sekaran (2009) added that the quantitative research design can produce consistent results when used with a descriptive research design. Several researchers have also identified the quantitative research design as the most suitable approach for investigating the individual opinions and the motives behind

the actions, behavior, and attitudes of respondents. Kumar (2011) and Atieno (2009) also identified the quantitative research design as the best scientific research method given its precise measurements via deductive approach and its employment of measurable data collection tools.

Babbie (2010) identified the quantitative research design as the most appropriate method to examine the relationship between dependent and independent variables. The academic success of programming students and other related factors can be examined by the quantitative research approach. The analysis hopefully answer the research questions and test the research hypotheses. According to Smith, (2012), questionnaires are better than most data collection instruments because of their inexpensiveness and anonymity.

3.3 Data Collection

The findings of most studies are generally supported by field data (Zikmund *et al*., 2010).

In this research, data has been collected as following:

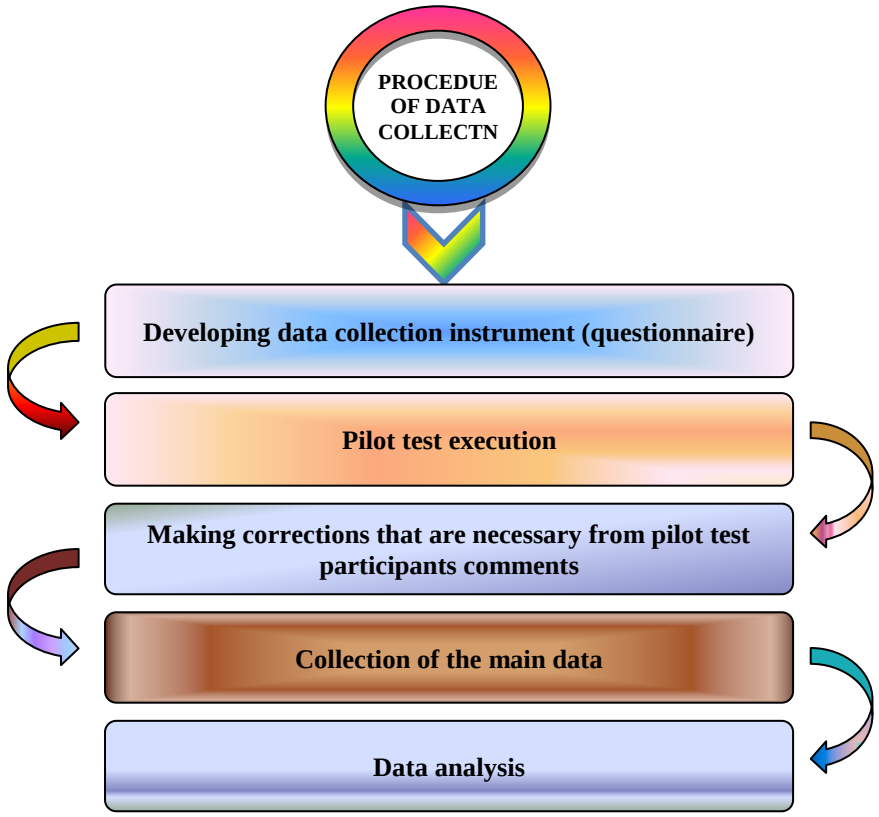

Figure 3. 2 Procedure of Data Collection

3.3.1 Population and Study Sample

This study was focused on the undergraduate and postgraduate programming students in IT and ICT of UUM. Several sampling methods are adopted to reveal the unidentified characteristics of the selected population.

This study adopts the simple random sampling technique in which all elements in the population are considered and such elements has an equal chance of being chosen as the subject in order for each aspect of the population to be represented in the sample (Zikmund *et al.*, 2010) and to provide accurate statistical descriptions of the population. According to Smith (2012) survey participants should be gathered in such a way that they are confined in one space. For example, a survey can be administered to students inside classrooms or to people in the middle of a seminar or a program. This method saves money for postage and ensures a high response rate given that the potential subjects will have no choice but to participate in the survey. The sample size for this study is determined through the rule of thumb, which states that the sample must include between 30 to 500 respondents (Sekaran & Bougie, 2010). According to the official letter from UUM the number of postgraduate and undergraduate students (IT and ICT programmes) of the years 2012-2013 is 566 students. A total of 286 students are selected to participate in this study based on Sekaran & Bougie (2010, P. 295). As mentioned by Notani (1998), studies on working behavior should focus on the general adult population than on the student population given that the former population are more experienced than the latter.

3.3.2 Research Instrument

A survey has been conducted to gather primary information on related factors. The use of questionnaire as the data gathering instrument is considered as efficient (Kumar, 2011). Furthermore, questionnaire that are self-administered having closed-ended questions.

The questionnaires for this study has adapted from Rogerson and Scott (2010), Bergin and Reilly (2005), Jenkins, (2001a), Jenkins (2001b), Gayle and Macklem (1990); Quenemoen, Thompson and Thurlow (2003), Barchard (2003) and Madden and Chambers (2002). as shown in the table 3.1:

Table 3.1 Questionnaires sources

No	Questionnaires of the factors	Adopted from
1	Academic success in programming	Kimberly and Barchard (2003)
2	Motivation	Jenkins (2001); Bergin and Reilly (2005)
3	Student Aptitude in Mathematics	Gayle and Macklem (1990); Quenemoen,Thompson and Thurlow(2003)
4	Fear	Rogerson and Scott (2010)
5	Experience with Other Programming Languages	Madden and Chambers (2002)
6	OOP Concepts	
7	Teaching Tools	
8	Course Evaluation	

For the instrument design, the questionnaire is divided into three parts: A, B and C. Part A asks questions related to the respondents demographic background which are gender, age group, course and previous programming experience. Part B contains items to measure the academic success in computer programming, while

Part C contains items measuring each of the elicited factors: motivation, fear, OOP concepts, teaching tools, course evaluation, and students' aptitude in Mathematics.

3.3.3 Pilot Test

A pilot study must be conducted before collecting data to validate the survey instrument (Bryman, 2004; Saunders *et al.*, 2003). A pilot study is conducted to determine if the questionnaire can be amended further for the respondents to understand and answer all questions with ease. Acceptable number of 30 respondents, were enough for the pilot study as the researcher was aiming only to examine to what extent the instrument was clear and therefore improve on it (Hair et al., 2010). A total of 40 questionnaires were distributed to UUM students to identify if these instruments are properly constructed and if the questions can be easily understood by the respondents. The students have been asked to answer these questionnaires and to provide some feedback with regard to the validity and clarity of the instrument.

3.4 Validity and Reliability

The validity and reliability of the developed measures must be ensured. The former refers to the capability of the instrument to assess the target items, whereas the latter refers to its consistency (Sekaran, 2003). According to Smith (2012, p. 5), *"the quality of a measurement procedure that provides repeatability and accuracy."*. The validity and reliability of the instrument has been analyzed after the pilot test. Smith (1991, p. 106) added the following: *"validity is defined as the degree to which the researcher has measured what he has set out to measure."*

Smith (2012) argued that validity only pertained to a particular instrument. However, a reliable measure may not be able to assess a specific item despite showing consistency. The reliability coefficient is expressed in terms of Cronbach's alpha.

An α of 0.70 to 0.80 is generally acceptable (Kaplan & Sacuzzo, 2008). The correlation between the dependent and independent variables must be estimated after ensuring the reliability of the measurements. However, ensuring the reliability of the measurements does not necessarily ensure their validity. The questionnaires can be validated by a group of expert judges (Kidder & Judd, 1986). Therefore, this study has sent the questionnaires to the expert who is (Dr Abdullah Al Swidi), where he has professional qualification in SPSS, SAS (Statistical Analysis Software), AMOS (SEM), Smart PLS (SEM), QM for Windows and Arena for Simulation. In addition he is member of the quantitative studies and development experts group, College of Arts and Sciences, University Utara Malaysia. Thereby, he reported that "*I have seen the questionnaire and the items used can serve the factors they were designed to measure*". Validity can be used to improve and evaluate the reliability of existing scales. Different procedures, such as factor analysis, can be used to establish construct validity (Zikmund *et al.*, 2010; Smith, 2012). Therefore, a pilot study was conducted to enhance the reliability and validity of the measures.

3.4.1 Face Validity

Face validity which is also called Content validation has to do with the testing respondents' comprehension of the items in the instrument. It refers to the transparency or relevance of a test as they appear to test participants Holden and Ronald (2010). This is very essential in this kind of research settings; it has been

done before proceeding to the main data collection stage, for the purpose of observing the mistakes in the instrument and to be corrected before going for the main data collection. For this purpose, each question of the instrument items was reframed and duplicated to examine if there could be any variation or misunderstanding to the response of any of the questions, this to ensure the research on how objective and authentic the gathered data are. Due to some constraints of getting feedback from the real candidates of face validity which were among the lecturers who has specialized in computing area, five PhD students specifically those that have defended their PhD thesis proposal were chosen for the face validity of the questionnaire. As Pallant (2011) and Zikmund *et al*. (2010) suggested, researchers are also among the suitable persons to be employed for face validity during the questionnaire development process.

3.5 Pilot Testing Result

The components of the pilot testing are the reliability testing of the items contained in the questionnaire and population distribution of the pilot study. The results shown in the following tables, and Appendix 2 (a-g) also shows the SPSS generated tables for all the variables studied.

3.5.1 Reliability Testing Results

The first of the pilot testing is the reliability testing of the items contained in the questionnaire. Table 3.2 presents the result of the reliability testing.

Table 3.2: Reliability Testing Result

Variable	Cronbach's Alpha	No of Items
Academic Success in Computer Programming	0.733	3
Motivation	0.901	8
Fear	0.776	12
Java concept	0.886	17
Teaching Tools	0.721	4
Course Evaluation	0.847	7
Aptitude in Mathematics	0.719	8

To achieve the reliability of the instrument, items of academic success in computer programming were seven items, after the reliability test, the Cronbach's Alpha was 0.4. There are four items has been dropped from academic success in computer programming based on option (scale if items deleted) in SPSS to enhance the Cronbach's Alpha. Thereby, the result as shown in the table 3.2 is accepted.

3.5.2 Population Distribution of the Pilot Study

This pilot study involves 38 males representing 95.0%, and 2 females represented by 5%. The result is shown in Table 3.2 below.

Table 3.3: Gender Distribution of the Pilot Study

		Frequency	Percent %
Valid	Male	38	95.0
	Female	2	5.0
	Total	40	100.0

Thirteen (13) out of the respondents are undergraduate students of Information technology (IT) and twenty-seven (27) are Master students (MSCIT/ICT), making 32.5% and 67.5% respectively. Table 3.3 shows the course level distribution.

Table 3.4: Course Level Distribution of the Pilot Study

		Frequency	Percent %
Valid	BSC IT	13	32.5
	MSc IT/ICT	27	67.5
	Total	40	100.0

From the respondents administered during the pilot testing phase, thirty (30) which is 75% have previous experience of programming, while ten (10) i.e. 25% do not. Table 3.4 shows the population distribution of respondents with previous experience with those without.

Table 3.5: Previous Programming Experience of the Pilot Study

		Frequency	Percent %
Valid	Yes	30	75.0
	No	10	25.0
	Total	40	100.0

3.6 Data Analysis

SPSS 20 has been used to analyze the data. Independent-Samples T test, correlation and regression are conducted as the descriptive analysis. In choosing the right statistic, need to consider a number of different factors. These include consideration of the type of question you wish to address, the type of items and scales that were included in your questionnaire, the nature of the data you have available for each of your variables and the assumptions that must be met for each of the different statistical techniques (Pallant, 2011). This study has two types of variables, continuous or ordinal variables which are (academic success in computer programming, teaching tools, fear, OOP concepts, motivation, course evaluation and students' aptitude in mathematics), and categorical variable which is (experience with other programming languages). Pearson correlation and linear regression has been used for the continuous or ordinal variables and Independent-Samples T test for the categorical variable.

According to Smith (2012), statistics and computers play a significant role in the research after the data collection procedure. The data analysis has been conducted to test the hypotheses and answer the research questions (Pallet, 2003).

The descriptive analysis examines the gathered responses and the distribution of the data to draw a possible conclusion. The table 3.6 showed the statistical analysis technique used.

Table 3.6: Statistical Analysis technique used

Hypothesis	**Statistical Analysis technique used**	**Justification**
H1,H3,H4,H5,H6,H7	➢ Pearson Correlation Coefficient ➢ Linear Regression	The type data and items were continuous or ordinal
H2	➢ Independent - Samples T Test	The type of data and items were categorical

3.7 Data Coding:

As mentioned above SPSS 20 has been used as statistical analysis technique for this study, one of the required steps is data coding which means represent each item of the questionnaires into code as shown in the tables below.

3.7.1 Data Coding for Academic Success in Computer Programming

Table 3.7: Academic Success in Computer Programming

No	**Items**	**Coding**
1	I do not get less than Bs in my programming related courses	AS1
2	I have won awards based on my programming proficiency	AS2
3	I have got scholarships/incentives based on my programming proficiency.	AS3

3.7.2 Data Coding for Motivation

Table 3.8: Motivation

No	Items	Coding
1	I want to be academically successful for my own satisfaction	MO1
2	I want to be academically successful to please my parents or family.	MO2
3	I want to be academically successful to please my teacher.	MO3
4	I want to be academically successful to get a good job.	MO4
5	I just want to be academically successful.	MO5
6	I want to be academically successful so as to be called a smart student	MO6
7	I want to be academically successful to get scholarship.	MO7
8	I want to be academically successful to get awards.	MO8

3.7.3 Data Coding for Fear

Table 3.9: Fear

No	Items	Coding
1	I have a problem associated with learning programming.	FE1
2	The word “programming” evokes the feeling of apprehension.	FE2
3	The word “programming” evokes the feeling of discomfort	FE3
4	I feel anxious during programming class	FE4
5	I feel panic during programming class	FE5
6	I feel stressed during programming class	FE6
7	I am very excited about learning programming	FE7
8	I am not distressed when I find any error in a program.	FE8
9	I have no problems with programming	FE9
10	I grasp programming concepts quite easily.	FE10
11	I do not achieve my blueprint through coding	FE11
12	My application takes much more time before it is successful	FE12

3.7.4 Data Coding for OOP Concepts

Table 3.10: OOP Concepts

No	Items	Coding
1	Syntax (e.g., language constructs and flow of control)	JC1
2	Using and defining methods	JC2
3	Using and defining arrays	JC3
4	String handling	JC4
5	Using I/O streams	JC5
6	File handling	JC6
7	Using and defining objects	JC7
8	Object-oriented programming (e.g., inheritance and polymorphism)	JC8
9	Exception handling	JC9
10	Using and writing applets	JC10
11	GUI programming	JC11
12	Multithreaded Programming	JC12
13	Using JDK library classes	JC13
14	Using JCreator	JC14
15	Using Netbeans	JC15
16	Using Eclipse	JC16
17	Java SDK development tools (e.g., appletviewer)	JC17

3.7.5 Data Coding for Teaching Tools

Table 3.11: Teaching Tools

No	Items	Coding
1	Lectures	TT1
2	Supervised labs	TT2
3	Tutorials	TT3
4	Assignments	TT4

3.7.6 Data Coding for Course Evaluation

Table 3.12: Course Evaluation

No	Items	Coding
1	I find the course useful.	CE1
2	I find the course enjoyable.	CE2
3	The recommended course textbook(s) are useful.	CE3
4	The course strikes good balance between theory and lab work.	CE4
5	The course provides hands-on practical work	CE5
6	The course provides employable knowledge	CE6
7	The course fairly touches all the core areas	CE7

3.7.7 Data Coding for Aptitude in Mathematics

Table 3.13: Aptitude in Mathematics

No	Items	Coding
1	I love dealing with figures than text	SA1
2	I am good at solving linear equations	SA2
3	I am good at solving exponential equations	SA3
4	I prefer expressing concepts using mathematics	SA4
5	I understand discrete mathematics	SA5
6	I do teach my classmates Mathematics	SA6
7	I understand mathematical representation of algorithm	SA7
8	I have a good knowledge of mathematics in data structure	SA8

3.8 Summary

The methodology of the research is presented in this chapter. Several procedures and justifications are incorporated in the methodology to fulfill the objectives and to answer the questions of the research. The research framework is also presented in this chapter.

This study uses a questionnaire as the primary data collection instrument. The questionnaire also has been piloted before conducting the main survey to test the validity and reliability of the measures (Chapter four). The survey data then be used to test the hypotheses and to fulfill the research objectives.

CHAPTER FOUR
DATA ANALYSIS AND RESEARCH FINDING

4.0 Introduction

This chapter addresses the data analysis stage of this study. It presents the data analysis process as done stage by stage in view of answering the earlier elicited research questions. This chapter presents the main data analysis consisting of data screening and cleaning, normalization of the data, homogeneity of the respondents, descriptive statistics, independent- T test, correlation and regression to duly answer the research questions and test the hypotheses.

4.1 Respondent Profile

The population distribution of the respondents is based on gender, course of study, age and previous experience in programming language. This is holistically presented in appendix 3.

For the gender distribution, out of the 286, 222 making 77.6% are males while 64 of 22.4% are the females. Table 4.1 presents the gender distribution of the respondents.

Table 4.1 Gender

		Frequency	**Percent**
Valid	Male	222	77.6
	Female	64	22.4
	Total	286	100.0

To depict the academic background of the respondents, the questionnaire administered inquires about the courses of the respondents. This is necessary so as to establish the compatibility of their academic background with the objective of the study. The respondents are exclusively drawn from BSc IT and MSc IT/ICT departments, and the distribution shows that 99 making 34.6% are BSc IT and 187 of 65.4% are MSc IT/ICT. Table 4.2 shows the course distribution of the respondents.

Table 4.2 Course

		Frequency	**Percent**
Valid	BSc IT	99	34.6
	MSc IT/ICT	187	65.4
	Total	286	100.0

The age distribution of the respondents is also reported. Out of the 286 respondents, 121 which is 42.3% are between 18-30 years old, 137; 47.9% are between 31-43 years old, while 28; 9.8% are 44 years and above. Table 4.3 presents the age distribution of the respondents.

Table 4.3 Age

		Frequency	**Percent**
Valid	18-30	121	42.3
	31-43	137	47.9
	44 and Above	28	9.8
	Total	286	100.0

The questionnaire administered asked to enquire about the previous programming experience of the respondents. It is important to note that previous programming experience is one of the factors that their relationships to academic success in computer programming are being studied. The profile shows that 218 which is 76.2% have previous experience in programming while 68; 23.8% do not have a previous programming experience. Table 4.4 shows the previous programming experience distribution.

Table 4.4 Experience

		Frequency	Percent
Valid	Yes	218	76.2
	No	68	23.8
	Total	286	100.0

4.2 Reliability Test

After the main data is gathered, a construct reliability test is done. The main data reliability test is to confirm the consistency of the construct scale and compare with the results gathered from the pilot testing. This is essential to establish the reliability of the study's instrument. Table 4.5 presents the verification –comparing the main with the pilot test results.

Table 4.5 Reliability Test

No	Variable	No. of Item	Pilot Test Cronbach's Alpha	Main Test Cronbach's Alpha
1	Academic Success in Computer Programming	3	0.733	0.705
2	Motivation	8	0.901	0.848
3	Fear	12	0.776	0.898
4	OOP concept	17	0.886	0.871
5	Teaching Tools	4	0.721	0.732
6	Course Evaluation	7	0.847	0.840
7	Aptitude in Mathematics	8	0.719	0.794

Assessing the table presented in 4.5 above with a comparative consideration to the values of Cronbach's Alpha generated for the pilot and main tests for each of the variables, it is observed that Academic Success in Computer Programming, Motivation and Course Evaluation recorded a lower value to what is obtained during the pilot test. However, values obtained at both ends are still greater than 0.7 which suggest the consistency of the items and the construct.

4.3 Data Screening

After the descriptive part of the data that concentrates on the population distribution is reported, data screening is performed on the gathered data sets so as to make it suitable for the inferential part of the data analysis. In this stance, the research questions and hypotheses testing can be confidently done. As Hair et al. (2010) posited stages of data screening to be executed before analyzing multivariate data specifically are missing data, detection of outliers, and normalization of the datasets.

4.3.1 Missing Data

All the items of the variables as gathered by this study are fed into the SPSS 20 for the detection of the missing values. Missing data are detected on items FE8, JC14, MO1, SA2 and SA3. These missing data were transformed appropriately using the missing value analysis procedure (Pallant, 2011).

4.3.2 Detection of Outliers

According to Tabachink and Fidel (2006), outliers are individual respondents of extreme scores on a specific variable among the set of variables in the questionnaires administered. It is also opined that it may distract the general result. Detection of outliers is done through the calculation of Mahalanobis distance for each respondent and then be compared with the Chi-Square with a significant error of 0.001. The Chi-Square is to be obtained from the general Chi-Square table using the number of items designed in the questionnaire as the determinant. This study has a total number of 59 items, making a critical value (X^2) of 98.34, and the maximum Mahalanobis distance (D^2) is 284.004. In totality, four respondents (coded 113, 132, 133 & 134) with D^2values 100.03, 230.10, 284.00 and 284.00 respectively are detected as outliers. Therefore, the sample size for the continuation of the data analysis becomes 282. Appendix 5 shows the output of the SPSS generated analysis process.

4.3.3 Normality of the Data

Data normality is necessary before proceeding to inferential analysis. In doing this, Skewness and Kurtosis are employed as measures for data normality (Pallant, 2011). Hair et al. (2010) posited that less than 2 z-skewness value is

appropriate for a sample size that is not big. Appendix 6 shows the descriptive statistics of the maximum and minimum values of the z-score that confirms the normality of the data used in this study.

4.3.4 Homogeneity of the Respondents

For a cogent reason, this study confirms the homogeneity of the respondents. The data collected for this study has students of undergraduate and postgraduate as its elements. This is done to ensure that there is no difference between the above two groups to be able for including them in the sample of this study. Therefore, the researcher conducts an independent t-test analysis on the data collected to confirm the insignificance of the course level to the recorded academic success value. The T-test result gives 12.80 and 12.61 as the mean value for BSc IT and MSc IT/ICT respectively. The results for the independent t-test are presented in Appendix 4.

The t-value of the result is 0.475 and the significant value (2-tailed) is 0.635 (greater than 0.05). This shows there is no significant variance in the mean value of academic success in computer programming for both the BSc IT and MSc IT/ICT students. This confirms that the sample elements of this study can be regarded as homogenous.

4.4 Testing the Research Hypotheses

As earlier posited, hypothesis testing of this study was exclusively be done using SPSS 20, however with varying statistical techniques determined by the peculiarity of the hypothesis to be tested. After the successful data screening and cleaning stage, varieties of statistical techniques are employed as found suitable for the research hypotheses. This study employs independent t-test, to compare the mean

score of the group with previous programming and the group without. Pearson Product-Moment correlation and Linear Regression are then used to find the strength and direction of the relationship between the variables, and the effect of each of the independent variables on the dependent variable (Sekaran, 2003; Hair *et al*, 2010).

H1: Teaching Tools can affect the students' academic success in programming course

The academic success in programming course being continuous and teaching tools effectiveness are firstly tested through Pearson product-moment correlation. The result showed that there is an insignificant and low positive relationship between teaching tool effectiveness and academic success in programming course.

For the regression analysis, although there is impact of teaching tools on academic success in programming course, yet, the effect is not highly significant as the value of adjusted R square indicate that the impact is quite weak.

The results for the correlation and regression are presented in table 4.6 and 4.7 respectively below. This points that the hypothesis: Teaching Tools affect the students' academic success in programming course is accepted.

Table 4.6: Correlation Result for Hypothesis 1

		Academic Success in Programming Course
Teaching Tools	Pearson Correlation	.040
	N	282

Table 4.7: Regression Result for Hypothesis 1

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.040[a]	.002	.002	3.151

a. Predictors: (Constant), TTT (Total sum of Teaching Tools Items)
b. Dependent Variable: TAS (Total sum of Academic Success Items)

H2: Previous Experience with other programming language can affect the academic success in programming course

The academic success in programming course being continuous and previous experience with programming language which is designed as a dichotomous variable (Yes or No) is firstly tested using Independent t-test. The descriptive statistics of the respondents showed that out of the 282 respondents, 215 answered 'Yes' to having previous experience in programming language, while 67 answered 'No'. The t-test result gives 12.69 as the mean value for the Yes group, and 12.63 for the No group. Though with a slight difference, it shows that the group with previous programming experience has a greater academic success mean value than those without.

The results for the independent t-test are presented in table 4.8 below. On the other hand, the t-value of the result is 0.139 (equal variance assumed) because the significant value of Levene's Test of Equality is 0.369, i.e. greater than 0.05. However, with the Significant value (2-tailed) of 0.889 (greater than 0.05), it shows that there is no significant variance in the mean value of the group's academic success in computer programming. This points that the hypothesis: Previous Experience with other programming language affects the academic success in programming course is not accepted.

Table 4.8: Independent T-test Result for Hypothesis 2

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
TAS	Equal variances assumed	.809	.369	.139	280	.889	.062	.441	-.807	.930
	Equal variances not assumed			.135	104.791	.893	.062	.456	-.843	.966

H3: Fear can affect the academic success in programming course

The academic success in programming course being continuous and fear are firstly tested through Pearson product-moment correlation. The result showed that there is an insignificant and low negative relationship between fear and academic success in programming course. This is explained by the correlation result given as r = -0.004, n = 282 and p >.05.

For the regression analysis, the value of R2is given as 0.000 as illustrated in table below. Thereby, there is no impact of fear on academic success in programming course.

The results for the correlation and regression are presented in table 4.9 and 4.10 respectively below. This points that the hypothesis: Fear affect the students' academic success in programming course is not accepted.

Table 4.9: Correlation Result for Hypothesis 3

		Academic Success in Programming Course
Fear	Pearson Correlation	-.004
	N	282

Table 4.10: Regression Result for Hypothesis 3

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	-.004[a]	.000	-.004	3.153

a. Predictors: (Constant), TFE (Total sum of Fear Items)
b. Dependent Variable: TAS (Total sum of Academic Success)

H4: OOP Concepts can affect the academic success in programming course

The academic success in programming course being continuous and OOP concepts are firstly tested through Pearson product-moment correlation. The result showed that there is a significant and high positive relationship between OOP concepts and academic success in programming course. This is explained by the correlation result given as $r = 0.817$, $n = 282$ and $p < .05$. The result shows approximately 81% variance in OOP concepts can be explained by 81% changes in the academic success in programming course variable.

For the regression analysis, the value of $R^2$is given as 0.668 which shows that 66% variance of the predictor (OOP concepts) explains 66% of the dependent variable; Academic success in programming course.

The results for the correlation and regression are presented in table 4.11 and 4.12 respectively below. This points that the hypothesis: OOP Concepts affect the academic success in programming course is accepted.

Table 4.11: Correlation Result for Hypothesis 4

		Academic Success in Programming Course
OOP Concepts	Pearson Correlation	**.817
	N	282

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

Table 4.12:Regression Result for Hypothesis 4

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.817[a]	.668	.667	1.817

a. Predictors: (Constant), TJC (Total Sum of OOP Concepts (OOP) items)
b. Dependent Variable: TAS (Total sum of Academic Success)

H5: Motivation can affect the academic success in programming course

The academic success in programming course being a continuous variable and motivation are firstly tested through Pearson product-moment correlation. The result showed that there is significant and high positive relationship between motivation and academic success in programming course. This is explained by the correlation result given as $r = 0.746$, $n = 282$ and $p < .05$. The result shows 76%

variance in motivation can be explained by 76% changes in the academic success in programming course variable.

For the regression analysis, the value of $R^2$is given as 0.556 which shows that 55% variance of the predictor (motivation) explains 55% of the dependent variable; Academic success in programming course.

The results for the correlation and regression are presented in table 4.13 and 4.14 respectively below. This points that the hypothesis: Motivation affects the academic success in programming course is accepted.

Table 4.13: Correlation Result for Hypothesis 5

		Academic Success in Programming Course
Motivation	Pearson Correlation	**.746
	N	282

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

Table 4.14:Regression Result for Hypothesis 5

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.746[a]	.556	.554	2.102

a. Predictors: (Constant), TMO (Total sum of Motivation Items)
b. Dependent Variable: TAS (Total sum of Academic Success)

H6: Course Evaluation can affect the academic success in programming course

The academic success in programming course being a continuous variable and course evaluation are firstly tested through Pearson product-moment correlation. The result showed that there is a significant and high positive relationship between course evaluation and academic success in programming course. This is explained by the correlation result given as $r = 0.602$, $n = 282$ and $p < .05$. The result shows 60% variance in course evaluation can be explained by 60% changes in the academic success in programming course variable.

For the regression analysis, the value of $R^2$is given as 0.362 which shows that approximately 36% variance of the predictor (course evaluation) explains 36% of the dependent variable; Academic success in programming course.

The results for the correlation and regression are presented in table 4.15 and 4.16 respectively below. This points that the hypothesis: Course Evaluation affects the academic success in programming course in programming course is accepted.

Table 4.15: Correlation Result for Hypothesis 6

		Academic Success in Programming Course
Course Evaluation	Pearson Correlation	**.602
	N	282

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

Table 4.16:Regression Result for Hypothesis 6

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.602[a]	.362	.360	2.518

a. Predictors: (Constant), TCE (Total sum of Course Evaluation Items)
b. Dependent Variable: TAS (Total sum of Academic Success)

H7: Mathematical Aptitude can affect the academic success in programming course

The academic success in programming course being a continuous variable and mathematical aptitude are firstly tested through Pearson product-moment correlation. The result showed that there is an insignificant and positive relationship between mathematical aptitude and academic success in programming course.

For the regression analysis, although there is impact of mathematical aptitude on academic success in programming course, yet the effect is not highly significant as the value of adjusted R square indicate that the impact is weak.

The results for the correlation and regression are presented in table 4.17 and 4.18 respectively below. This points that the hypothesis: Mathematical aptitude affects the students' academic success in programming course is accepted.

Table 4.17: Correlation Result for Hypothesis 7

		Academic Success in Programming Course
Mathematical Aptitude	Pearson Correlation	.082
	N	282

Table 4.18:Regression Result for Hypothesis 7

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.082[a]	.007	.003	3.143

a. Predictors: (Constant), TSA (Total sum of Students' Aptitude in Mathematics)
b. Dependent Variable: TAS (Total sum of Academic Success)

4.5 Summary

The findings of the study which contains the respondents' profile, population distribution, data cleaning and screening stage, the hypotheses testing using Pearson Correlation and Linear Regression are presented in this chapter. In view of this, the hypotheses as tested by the study are brought to the fore with appropriate answers to the research questions elicited. At the end, hypotheses 2 and 3 are the ones that are not accepted. The following chapter ends this report of this study by extensively discussing the findings in view of its position and relevance among previous related studies.

CHAPTER FIVE
CONCLUSION AND RECOMINDATION

5.0 Introduction

This chapter concludes this study. It entails the discussion of the research findings, interpretation of the entire result of this study and the accompanying discussion. It argues the position of the findings of this study amidst the previous studies' findings. Finally, it points to the accomplishment of the study's objectives and establishes its practical and theoretical contribution. (Figure 5.1) showed the student success model based on the finding of this study, discussion of each factor in the following sections.

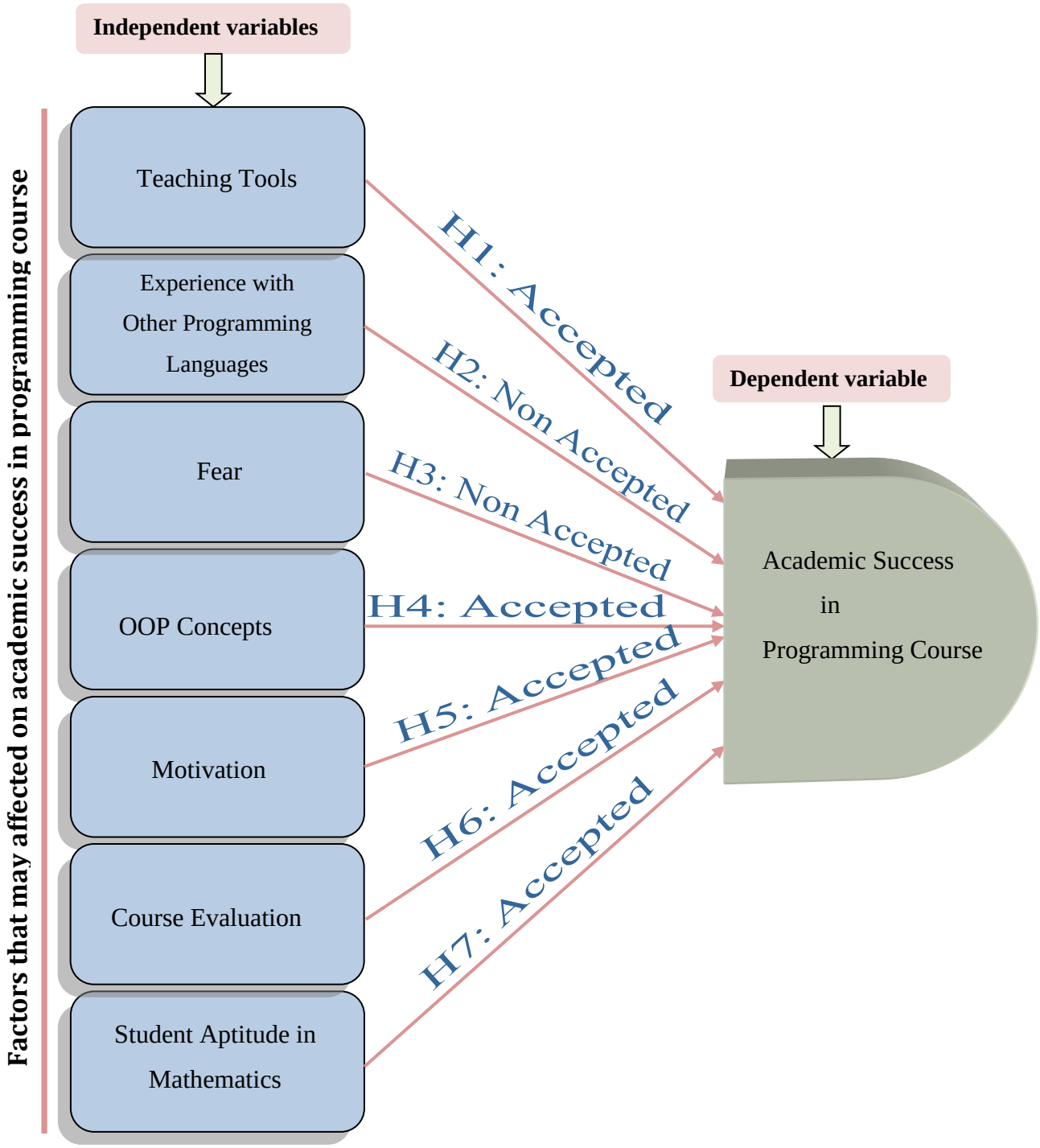

Figure 5. 1 Student Success Model

5.1 Discussion

The discussion of the findings of this study starts from the descriptive analysis done. Firstly, out of the total 286 respondents studied, 222 were males, while 64 were females. This result does not have any analytical importance because gender is not involved in the determining variables of this study. Also, the course of study distribution of the respondents reveals 99 respondents to be of BIT, while 187 respondents are MSc IT or/and MSc ICT. This study deals with this distribution as being homogenous since they are all students of IT irrespective of their course of study. The age distribution shows that 121 respondents are between 18-30 years, 137 are between 31-43 years, and 28 are 44 years and above. Age as a variable does not have any analytical importance in this study also. However, since previous experience in programming course is one of the studied variables, an item was designed to enquire this from the respondents, and serve as classifying guide into two groups; namely, respondents with previous experience, and respondents without previous experience. The experience distribution of the respondents shows that 218 respondents have previous experience in programming language, while 68 do not have. The result of the independent t-test conducted to test if previous experience in programming course affects success in programming success is discussed under section 5.1.2. more details in the following sections.

5.1.1 Hypothesis 1: Teaching tools can affect the students' academic success in programming course

This study found that teaching tool is related to academic success in programming, though with a low variance, and it also has weak impact on academic success in programming course. This result agrees with findings from studies like Brusilovsky, Calabrese, Hvorecky, Kouchnirenko, and Miller (1997), Cooper, Dann, and Pausch (2003) and Milne and Rowe (2002), and Ulloa (1980), without an explicit information about the effect relationship. In their cases, e-learning materials and tools like program visualization tool and animations of computational elements are the examples of teaching tools studied. Other studies like Jenkins (2002) claimed teaching tools affects academic success in programming course when they are employed with peers teaching method.

5.1.2 Hypothesis 2: Experience with other programming languages can affect the academic success of programming students

The result of this hypotheses shows that the Levene significant (2-tailed) value is greater than 0.05, contrary to the expected less than 0.05 to prove that there is significant variation in the mean value of academic success in programming course for group with previous experience is higher than that of no experience, so as to accept the hypothesis. This however shows that previous experience does not affect academic success in programming course. This result is in disagreement with Thompson (1995). Other related studies (Armoni *et al*., 2012; Burton & Bruhn, 2003) were just conceptual arguments to support that previous experience affects academic success in programming course, without any empirical study. This means the finding of this study can be placed for further empirical findings.

5.1.3 Hypothesis 3: Fear can affect the academic success of programming students

This study found that fear and academic success in programming course are negatively related. This implies that increase in fear leads to decrease in academic success in programming course, and vice versa. Also recorded, with the regression result is that fear does not have effect on academic success of programming course. To the best knowledge of the researcher, no empirical study has been instituted in this direction. Robins *et al*. (2003a) and Mead *et al*. (2006) only reported that the fear of computer science and information technology courses have been recorded to be responsible for high rate of attrition in these courses.

5.1.4 Hypothesis 4: OOP concepts can affect the academic success of programming students

The finding of this study reveals that having a grasp of OOP concepts is related to students' academic success. The regression analysis also showed that it affects academic success of programming course. Abstraction (Sebesta, 2004) and polymorphism (Weiss, 2000;Rosson & Alpert, 1990) are the key leading OOP concepts identified. There are no empirical findings to support this proposition extensively, however, reports of Object Management Group (2003) and Alphonce and Ventura (2002) suggested that the use of diagrammatic representations can help students improve their understanding of OOP, and UML or other analogous notations are also recommended.

5.1.5 Hypothesis 5: Motivation can affect the academic success of programming students.

This study found that motivation is highly related to academic success of programming course. The regression analysis also showed that it affects academic success of programming course. This finding aligns with studies of Gomes and Mendes (2007), Jenkins (2002), Robins *et al*. (2003b) and Simon *et al*. (2006). While Carbone, Hurst, Mitchell, and Gunstone (2009) explanation was basically on the categorization of motivation into externally motivated (i.e., passing the course) or achievement-motivated (i.e., obtaining higher marks), Helme and Clarke (2009) stated that students need motivation (the will to learn) and skills (capability) in order to be successful in their respective fields. Therefore, it is recommended that programming language educators and instructors should find ways of motivating the students, so as to enhance their performance.

5.1.6 Hypothesis 6: Course evaluation (which include: lectures, laboratory work, tutorials, and assignments) can affect the academic success of programming students

This study found that course evaluation is related to academic success of programming course and the regression result also showed that it affects it. To the best knowledge of the researcher, no study has ever empirically tested this relationship. Souza *et al.*, (2008) was said to have observed that programming students learning activities affects the instructors' attention to course evaluation. (Jenkins, 2002) reported that instructors may not be paying attention to their students and not addressing their learning styles had been posing additional problems.

5.1.7 Hypothesis 7: Mathematical aptitude can affect the academic success of programming students

This study found that mathematical aptitude is related to academic success of programming course. The regression analysis also showed that it affects academic success of programming course. Findings of Patil (2009) stated that students admitted with passing level in mathematics aptitude show significant effects in development factors as well as programming skills, and Jenkins (2002) also argued that students who find programming difficult are simply and solely those for whom programming is difficult. Although none of their positions is empirically backed, it points to the fact the result of this study align with their propositions on the effect of mathematical aptitude on the academic success in programming courses. This can be further understood considering that Mathematics is a prerequisite for admission to study Computer science in the university. Also, there are courses of mathematical background that usually helps in the understanding of programming logic. These explain why mathematical aptitudes could have effect on academic success in programming language.

5.2 Conclusion

With the highlights from the literature review and the outlined previous studies' findings that are related to the objectives of this study, the data analysis duly accomplished the hypotheses testing stage, though with varying results. From the result of the correlation testing as shown previously, this study found that all the factors studied, i.e. teaching tools, previous programming experience, fear, OOP concepts, motivation, course evaluation, and mathematical aptitudes are related with academic success in programming course, with fear as the only variable that is negatively related. The regression analysis to investigate the effect of the elicited on independent variables on the dependent variable and independent t-test revealed that previous experience and fear do not affect academic success in programming. This made hypotheses 1, 4, 5, 6 and 7as the accepted ones, while hypotheses 2 and 3 are not accepted.

From the earlier posed research questions and their corresponding research objectives, literature review reveals teaching tools, previous programming experience, fear, OOP concepts, motivation, course evaluation, and mathematical aptitudes as factors that may affect academic success of students in computer programming. These elicited factors amount to the development of computer programming students' success model presented in figure 2.3. Thereafter, the evaluation of this model is done through the outlined data analysis processes and hypotheses testing methods. The findings of the study however state that the earlier listed factors affect academic success in programming course, but previous experience and fear do not. H2 and H3 are not accepted probably because most of the participants' age were 31 years old and above, which means that the fear does not really contribute as the main predictor on academic success in computer

programming. In addition, most of them have previous experience with other programming languages. However, they still struggle for getting high performance in programming course. Consequently, there is no different between the group with or without previous experience.

In conclusion, empirical findings presented by this study established the effect relationship between teaching tools, OOP concepts, motivation, course evaluation, and mathematical aptitudes and fear with academic success in programming.

5.3 Contribution of the Study

Theoretically, this study has presented an updated success model for measuring students' academic success. This model as presented in figure 5.1 shows the antecedents factors that lead to academic success. It is adaptable and adoptable for future studies investigating academic success generally. The model has been able to be supported by findings of some previous studies, and equally presented findings that are new, therefore require further studies. Practically, the findings of this study, most especially the factors identified by the empirical findings can be implemented so as to improve the academic performance of student in programming courses.

This findings presented by this study will guide policy makers, educators and IT training in academics and industry in formulating policies, education curriculum and teaching modules. In such case, the factors identified to have effect on academic success of programming language will be taken into consideration in policy implementation so as to enhance students' academic success. It is opined that doing this will positively contribute to the students' performance improvement in computer programming courses.

5.4 Limitations of the Study

This study focuses solely on the student's perceptions to identify the factors that lead to academic success instead of their performance which can also contribute as a moderate variable. While, teacher's perceptions also can aim to generate different factors that impact on the learning process. Unfortunately, due the time constraints and financial aspects led to a narrowing of the scope of this research. This was confirmed by Sekaran (2003) and Creswell (2009), where referred that, the time, cost and willingness of the participants important criteria for researcher to identify the scope.

5.5 Recommendation for Future Study

Since research is naturally in continuum, the end of a study signifies the continuation of another one. This study, as instructing as its findings are further suggested future study that will investigate the possible interplay of some variables as mediators and/or moderators in the cause of academic success in computer programming course. Also recommended is the employment of more sophisticated statistical tool like structural equation model using AMOS or PLS-R. Grounding the findings through more sophisticated tools will also add to the strength of the findings.

5.6 Summary

This chapter is the end of this study's report. It concludes the findings presented by this study as detailed and duly marshaled to address the objective of this study. The comparison of the findings presented by this study with other related previous studies showed that this study has been able to contribute both theoretically and practically. Recommendations for further studies are made and the areas are duly suggested.

REFERENCES

Alphonce, C., & Ventura, P. (2002). Object Orientation in CS1-CS2 by Design. *ACM SIGCSE Bulletin*, *34*(3). doi:10.1145/637610.544437.

Armoni, M., Gordon, M., & Harel, D. (2012). The Effect of Previous Programming Experience on the Learning of Scenario-Based Programming. *In Proceedings of the 12th Koli Calling International Conference on Computing Education Research.ACM*, 151–159.

Astrachan, & T. Selby, J. U. (2006). An object-oriented, apprenticeship approach to data structures using simulation. *In Frontiers in Education Conference, 1996. FIE'96. 26th Annual Conference., Proceedings of* (Vol. 1, pp. 130-134). IEEE.

Atieno, O. (2009). An Analysis of the Strengths and Limitations of Qualitative and Quantitative Research Paradigms. *Problems of Education in the 21st century*, *13*, 13–18.

Babbie, E. (2010). *The Practice of Social Research,12th Edition.* Wadsworth Cengage Learning, USA.

Bailie, F. (2003). Objects First - Does It Work ? *Journal of Computing in Small Colleges*, *19*(2), 303–305.

Bennedsen, J., & Caspersen, M. E. (2006). Abstraction ability as an indicator of success for learning object-oriented programming? *ACM SIGCSE Bulletin*, *38*(2), 39. doi:10.1145/1138403.1138430.

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin*, *39*(2), 32. doi:10.1145/1272848.1272879.

Bennett, G., Fisher, M., & Lees, B. (2011). Object-Oriented Programming with Objective-c. *In Objective-C for Absolute Beginners*, 87–102.

Bergin, S., & Reilly, R. (2005a). Programming : Factors that Influence Success. *In ACM SIGCSE Bulletin*, 411–415.

Bergin, S., & Reilly, R. (2005b). The influence of motivation and comfort-level on learning to program. *In Proceedings of the PPIG*, (June), 293–304.

Biddle, R., & Tempero, E. (1998). Java Pitfalls for Beginners. *SIGCSE Bulletin*, *30*(2), 48–52.

Biju, S. M. (2013). Difficulties in understanding object oriented programming concepts. *In Innovations and Advances in Computer, Information, Systems Sciences, and Engineering (pp. 319-326).* Springer New York.

Black, A. P., Bruce, K. B., Homer, M., Noble, J., Yannow, R., Weishaupt, A., & Hazlitt, W. (2013). Seeking Grace : A new object-oriented language for novices. *In Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 129–134.

Blackwell, A. F. (1996). Metacognitive theories of visual programming: what do we think we are doing? *Proceedings IEEE Symposium on Visual Languages*, 240–246. doi:10.1109/VL.1996.545293.

Bougie, R., & Sekaran, U. (2010). *Research methods for business (5th ed.).* West Sussex, United Kingdom: John Wiley & Sons Ltd.

Bruce, C., Buckingham, L., Hynd, J., Mcmahon, C., Roggenkamp, M., & Stoodley, I. (2004). Ways of Experiencing the Act of Learning to Program : A Phenomenographic Study of Introductory Programming Students at University. *Journal of Information Technology Education, 3, 143-160.* Retrieved from http://www.jite.org/documents/Vol3/v3p143-160-121.pdf on 3rd February, 2014.

Bruce, K. B. (2005). Controversy on how to teach CS 1: A discussion on the SIGCSE-members mailing list. *SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education)*, *37*(2), 111–117.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., & Miller, P. (1997). Mini-languages: a way to learn programming principles. *Education and Information Technologies*, *83*, 65–83.

Bryman, A. (2004). *Social research methods.* New York: Oxford University Press Inc.

Burkhardt, J., Détienne, F., Wiedenbeck, S., Voluceau, D. De, & Chesnay, L. (1997). Mental Representations Constructed by Experts and Novices in Object-Oriented Program Comprehension. *In Human-Computer Interaction INTERACT'97, pp. 339-346).* Springer US., 339–346.

Burns, R. B. (1997). *"Introduction to Research Methods in Education,"* Melbourne, Victoria: Longman Cheshire.

Burton, P. J., & Bruhn, R. E. (2003). Teaching Programming in the OOP, *ERA*, *35*(2), 111–114.

Butler, M., & Morgan, M. (2007). Learning challenges faced by novice programming students studying high level and low feedback concepts. *In Proceedings of Ascilite*, 99–107.

Byrne, M. D., Catrambone, R., & Stasko, J. T. (1999). Evaluating animations as student aids in learning computer algorithms. *Computers & Education*, *33*(4), 253–278. doi:10.1016/S0360-1315(99)00023-8.

Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE '01*, 49–52. doi:10.1145/377435.377467.

Carbone, A., & Hurst, J. (2009). An E xploration of I nternal F actors I nfluencing S tudent L earning of P rogramming. *Proceedings of the Eleventh Australasian Conference on Computing Education*, *95*(Ace), 25–34.

Carbone, A., Hurst, J., Mitchell, I., & Gunstone, D. (2009). An exploration of internal factors influencing student learning of programming. *In Proceedings of the Eleventh Australasian Conference on Computing Education . Australian Computer Society, Inc.., 95*, 25–34.

Carlisle. (2009). Raptor: A Visual Programming Environment For Teaching Object-Oriented Programming*. *Journal of Computing Sciences in Colleges*, *24*, 275–281.

Caspersen, M. E., Kölling, M., Ct, K., & Beck, K. (2006). A Novice ' s Process of Object-Oriented Programming. *Companion to the 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages, and applications,ACM*, 892–900.

Cheney, E. E. W., & Kincaid, D. R. (2012). *Numerical mathematics and computing. C*engage Learning.

Clark, D., MacNish, C., & Royle, G. F. (1998). Java as a teaching language—opportunities, pitfalls and solutions. *In Proceedings of the 3rd Australasian Conference on Computer Science education,ACM*, 173–179.

Close, D. Kopec, and J. A. (2000). CS1: Perspectives on Programming Languages and the Breadth-First Approach. *In Proceedings of the 5th Annual CCSC Northeastern Conference on Computing in Small Colleges*, 1–7.

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. *ACM SIGCSE Bulletin, 35*(1), 191. doi:10.1145/792548.611966

Costa, C. J., Iscte, A., & Pierce, R. (2009). Evaluating Information Sources for Computer Programming Learning and Problem Solving. *In Proceedings of the 9th WSEAS International Conference on APPLIED COMPUTER SCIENCE*, 218–223.

Crawford, S. & Boese, E. (2006). ActionScript: a gentle introduction to programming. *Journal of Computing Sciences in Colleges*, *21*(3), 156–168.

Dehnadi, S., & Bornat, R. (2006). *The Camel Has Two Humps* (working title). Middlesex University, UK. 1–21.

Eckerdal, A., Box, P. O., & Thun, M. (2005). Novice Java programmers' conceptions of object and class, and variation theory. *In ACM SIGCSE Bulletin* (Vol. 37, No. 3, pp. 89-93).

Eckerdal. (2006). Novice Students ' Learning of Object-Oriented Programming.

El-Zakhem, I., & Melki, A. (2013). Difficulities In Learning Programming LanguagesAmong Freshman Students. *INTED2013 Proceedings*, 1202–1206.

Entwistle, N. (1998). Motivation and approaches to learning: motivation and conceptions of teaching. *In: Brown, S., Armstrong, S., Thompson, G. (Eds.), Motivating Students. Kogan Page, London, United Kingdom.*

Floyd, B., & London, R. (1970). I . Notes on Structured Programming: *Technological University Eindhoven.*

Garner, S. K. (2001). Cognitive load reduction in problem solving domains.

Garner, S., Haden, P., & Robins, A. (2005). My Program is Correct But it Doesn ' t Run: A Preliminary Investigation of Novice Programmers ' Problems. *In Proceedings of the 7th Australasian Conference on Computing Education*, *42*, 173–180.

Garrido, J. M. (2004). Object-Oriented Programming: From Problem Solving to Java. *Firewall Media.*

Georgatos (2008). How applicable is Python as first computer language for teaching programming in a pre-university educational environment, from a teacher's point of view?. *arXiv preprint arXiv:0809.1437.*

Gomes, A., & Mendes, A. J. (2007). Learning to program - difficulties and solutions. *In International Conference on Engineering Education–ICEE*.

Graf, S., Lan, C. H., & Liu, T.-C. (2009). Investigations about the Effects and Effectiveness of Adaptivity for Students with Different Learning Styles. *2009 Ninth IEEE International Conference on Advanced Learning Technologies*, 415–419. doi:10.1109/ICALT.2009.135

Gries, D. (1974). What should we teach in an introductory programming course? *SIGCSE '74: Proceedings of the Fourth SIGCSE Technical Symposium on Computer Science Education*.

Grinnell, R. jr. (ed.). (1993). *Social Work, Research and Evaluation", (*4th ed), Illinois, F.E Peacock Publishers.

Grogono. (1989). Comments, assertions and pragmas. *ACM SIGPLAN Notices,*, *24*(3), 79–84.

Gross, P. & Powers, K. (2005). Evaluating assessments of novice programming environments. *Proceedings of the 2005 International Workshop on Computing Education Research ICER '05*, 99–110.

Guthrie, R., Yakura, E., & Soe, L. (2011). How Did Mathematics and Accounting Get So Many Women Majors ? What Can IT Disciplines Learn?, *1886*(909), 15–19.

Hadjerrouit, S. (1998). Java as First Programming Language: A Critical Evaluation, *30*(2).

Hadjerrouit, S. (1998). Java as First Programming Language: A Critical Evaluation,. *ACM SIGCSE Bulletin*, *30*(2), 43–47.

Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E. &Tattam, R. L. (2010). *Multivariate Data Analysis* (7th ed.), New Jersey: Pearson Education Inc.

Hardy, C., Heeler, P., & Brooks, D. (2006). Are High School Graduates Technologically Ready For Post-Secondary Education? *Journal of Computing Sciences in Colleges*, *21*(4), 52–60.

Hasan, N. M. and Y. (2001). Challenges in teaching java technology. *Informing Sci.*, 365–371.

Havenga, H. M. (2006). *An Investigation Of Students ' Knowledge , Skills And Strategies During Problem Solving In Object-Oriented Programming.*

Havenga, M., Mentz, E., & De Villiers, R. (2008). Knowledge, skills and strategies for successful object-oriented programming: a proposed learning repertoire. *South African Computer Journal*, *42*, 1–8.

Helme & Clarke. (2009). Identifying cognitive engagement in the mathematics classroom. *Mathematics Education Research Journal*, *13*, 133–153.

Helme, S. U. E., & Clarke, D. (2001). Identifying cognitive engagement in the mathematics classroom. *Mathematics Education Research Journal*, 131–153.

Henderson, R., & Zorn, B. (1994). A Comparison of Object-oriented Programming in Four Modern Languages, *24*(June), 1077–1095.

Herman, N. S., & Salam, S. B. (2011). A Study of Tracing and Writing Performance of Novice. *In Software Engineering and Computer Systems . Springer Berlin Heidelberg.*, 557–570.

Holden, Ronald B. (2010). "Face validity". In Weiner, Irving B.; Craighead, W. Edward. *The Corsini Encyclopedia of Psychology* (4th ed.). Hoboken, NJ: Wiley. pp. 637–638. ISBN 978-0-470-17024-3.

Holland, S., Griffiths, R., Woodman, M., Hall, W., Keynes, M., & Kingdom, U. (1997). Avoiding Object Misconceptions. *In ACM SIGCSE Bulletin*, *29*, 131–134.

Hossein, S. (2007). *Response modeling in direct marketing. Master thesis, Department of business administration and social science,* University of Technology, Iran.

Howles, T. (2007). Preliminary Results Of A Longitudinal Study Of Computer Science Student Trends, Behaviors and Preferences. *Journal of Computing Sciences in Colleges*, *22*, 18–27.

Jenkins, T. (2001a). Teaching programming–A journey from teacher to motivator. *Paper Presented at the The 2nd Annual Conference of the LSTN*, Center for Information and Computer Science.

Jenkins, T. (2001b). The motivation of students of programming. *ACM SIGCSE Bulletin*, *33*(3), 53–56.

Jenkins, T. (2002). On the difficulty of learning to program. *In Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, 53–58.

Kaplan, R., & Sacuzzo, D. (2008). *Psychological Testing: Principles, applications and Issues, Brooks Cole. Pacific Grove, CA.*

Kerlinger, F. N. (1986). *Foundation of Behavioral Research, 3th edition. New York: Holt, Rinehart & Winston.*

Kidder, L. H., & Judd, C. M. (1986). *Research methods in social relations (5th ed.). New York: Holt, Rinehart and Winston*.

Kimberly A. Barchard. (2003). Does Emotional Intelligence Assist in the Prediction of Academic Success? *Educational and Psychological Measurement*; 6(3), 840-858.

Kölling, M. (1999). The problem of teaching object-oriented programming. *In ACM Sigplan Notices*, *11*(8), 8–15.

Kölling. (2003). The BlueJ system and its pedagogy 1. *Computer Science Education, 13*(4),1–12.

Kumar, R. (2011). *Research methodology: A step-by-step guide for beginners (3rd ed.).*Thousand Oaks, CA: SagePublications Inc.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14. doi:10.1145/1151954.1067453

Lakshman, M., Sinha, L., Biswas, M., Charles, M., & Arora, N. K. (2000). Quantitative Vs qualitative research methods. *The Indian Journal of Pediatrics*, *67*(5), 369–377. doi:10.1007/BF02820690

Li, T., Liu, W., Mao, X., & Zhou, H. (2013). Introduction to Programming : Science or Art? *In Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, 4503.

Macklem, Gayle L. (1990). Measuring aptitude. *Practical Assessment, Research & Evaluation*, 2(5).

Madden, M., & Chambers, D. (2002a). Evaluation of Student Attitudes to Learning the Java Language. *In Proceedings of the inaugural conference on the Principles and Practice of programming, 2002 and Proceedings of the second workshop on Intermediate representation engineering for virtual machines*, (pp. 125-130). National University of Ireland.

Madsen, O. L., & Møller-Pedersen, B. (1988). What object-oriented programming may be-and what it does not have to be. *In ECOOP'88 European Conference on Object-Oriented Programming (pp. 1-20). Springer Berlin Heidelberg.*

Mason, R. (2012). Designing introductory programming courses: the role of cognitive load.

Mason, R., Cooper, G., & Raadt, M. De. (2012). Trends in Introductory Programming Courses in Australian Universities – Languages , Environments and Pedagogy. *Computing Education Conference (ACE2012)*, *123*, 33–42.

Matravers, J. (2011). Introduction to computer systems architecture and programming. *University of London*.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B., & Al., E. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education)*, *33*(4), 125–180.

Mcgettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., & Mander, K. (2005). Grand Challenges Education. *The Computer Journal*, *48*(1), 49–52.

McGettrick. (2005). Grand challenges in computing: Education--A summary. *The Computer Journal,*, *48*(1), 42–48.

Mead, J., Gray, S., Hamer, J., James, R., Sorva, J., Clair, C. St., & Thomas, L. (2006). A cognitive approach to identifying measurable milestones for programming skill acquisition. *In ACM SIGCSE Bulletin*, *38*, 182–194. doi:10.1145/1189215.1189185

Mills, H. D. (1972). *Mathematical Foundations for Structured Programming*.

Milne, I., & Rowe, G. (2002). Difficulties in Learning and Teaching Programming — Views of Students and Tutors. *Education and Information Technologies*, 55–66.

Moderator, O. A., Koffman, E., Kölling, M., & Reges, S. (2005). Resolved : Objects Early Has Failed. *In ACM SIGCSE Bulletin*, *37*, 451–452.

Mody, R. P. (1991). C in education and software engineering. *ACM SIGCSE Bulletin*, *23*(3), 45–56. doi:10.1145/126459.126471

Mow, I. T. C. (2008). Issues and difficulties in teaching novice computer programming. *In Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education* (pp. 199-204). *Springer Netherlands.*

Nikishkov, G. P., Nikishkov, Y. G., & Savchenko, V. V. (2003). Comparison of C and Java Performance in Finite Element Computations. *Computers & Structures*, *81*(24), 2401–2408.

Notani, A. S. (1998). Moderators of perceived behavioural control's predictiveness in the theory of planned behaviour: A meta-analysis. *Journal of Consumer Psychology*, *7*(3), 247–271.

Object Management Group. (2003). Unified Modeling Language Specification. *Version 1.5 March 2003, Doc. Number formal/03-03-01*.

Pair, C. (1993). Programming, programming languages and programming methods. *Psychology of Programming*, 9-19.

Pallant, J. (2003). *SPSS survival manual: A step by step guide to data analysis using SPSS for Windows (Version 10)*. Australia: Allen & Unwin.

Pallant, J. (2011). *For the SPSS Survival Manual website* , go to www.allenandunwin.com/spss This is what readers from around the world say about the SPSS Survival Manual.

Patil, P. S. P. (2009). The effect of developments in student attributes on success in Programming of management students. *In Education Technology and Computer, 2009. ICETC'09. International Conference on IEEE.*, 191–193. doi:10.1109/ICETC.2009.35

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Uni, J. M., … Paterson, J. (2007). A Survey of Literature on the Teaching of Introductory Programming. *In ACM SIGCSE Bulletin*, *39*(4), 204–223.

Pejcinovic, Holtzman, M., Chrzanowska-Jeske, & W. (2013). Just because we teach it does not mean they use it: Case of programming skills. *In Frontiers in Education Conference*, 1287–1289.

Poo, D. C., Kiong, D. B. K., & Ashok, S. (2007). *Object-Oriented Programming and Java* (2nd Editio). Springer.

Powers, K., Gross, P., Cooper, S., McNally, M., Goldman, K. J., Proulx, V. & Carlisle, M. (2006). Tools for teaching introductory programming: what works? *ACM SIGCSE Bulletin , Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education SIGCSE '06*, *38*(1), 560–561.

Quenemoen, R., Thompson, S. & Thurlow, M. (2003). *Measuring academic achievement of students with significant cognitive disabilities: Building understanding of alternate assessment scoring criteria*(Synthesis Report 50). Minneapolis, MN: University of Minnesota, National Center on Educational Outcomes.

Reis, C., Tx, H., & Cartwright, R. (2004). Taming a Professional IDE for the Classroom. *In ACM SIGCSE Bulletin*, *36*, 156–160.

Renumol, V. G., Jayaprakash, S., & Janakiram, D. (2009). Classification of Cognitive Difficulties of Students to Learn Computer Programming. *Indian Institute of Technology, India.*

Rinard, M. C., Scales, D. J., & Lam, M. S. (1993). Jade: A High-Level , *Language for Parallel Programming,26*, 28–28.

Robins, A., Rountree, J., & Rountree, N. (2003a). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, *13*(2), 137–172. doi:10.1076/csed.13.2.137.14200

Robins, A., Rountree, J., & Rountree, N. (2003b). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, *13*(2), 137–172. doi:10.1076/csed.13.2.137.14200

Rogerson, C., & Scott, E. (2010). The Fear Factor: How It Affects Students Learning to Program in a Tertiary Environment. *Journal of Information Technology Education*.

Rosson, M. B., & Alpert, S. R. (1990). The cognitive consequences of object-oriented design. *Human-Computer Interaction*, *5*(4), 345–379.

Rowe, I. M. and G. (2002). Difficulties in learning and teaching programming-Views of students and tutors. *J. Edu. & Info. Tech.*, *7*, 55–66.

Sajaniemi, J., & Kuittinen, M. (2003). Program animation based on the roles of variables. *Proceedings of the 2003 ACM Symposium on Software Visualization - SoftVis '03*, 7. doi:10.1145/774834.774835

Satzinger, J.W. & Ørvik, T. U. (2001). *The Object-Oriented Approach. Concepts, System Development, and Modeling with UML.*

Saunders, M., Lewis, P., & Thronhill, A. (2003). *Research method for business students (3 ed). England:* Person Education Limited.

Schach, S. R. (2005). *Object-Oriented and Classical Software Engineering. (6th ed.). Boston: McGraw-Hill.*

Schreiner, w. (2011). *Introduction to Programming* (pp. 1–156).

Schulte, C., & Bennedsen, J. (2006). What do teachers teach in introductory programming? *In Proceedings of the Second International Workshop on Computing Education Research ACM*, p 17–28.

Sebesta, R. W. (2004). *Concepts of Programming Languages. (6th ed.).* Boston: Pearson Addison Wesley.

Sekaran, U. (2003). *Research Methods for Business: A skill building approach.* John Wiley and Sons Inc., New York.

Sekaran, U., & Bougie, R. (2009). *Research methods for business: A skill building approach.* Wiley: London.

Sekaran, U., & Bougie, R. (2010). *Research methods for business: A skill building approach.* Wiley.

Sharp & Schultz. (2013). An Exploratory Study of the use of Video as an Instructional Tool in an Introductory C# Programming Course. *Information Systems Education, 11*(6).

Sicilia, M.-Á. (2006). Strategies for teaching object-oriented concepts with Java. *Computer Science Education*, *16*(1), 1–18. doi:10.1080/08993400500344431

Simon, S., Fincher, S., Robins, A., Baker, B, Box, I., Cutts, Q., de Raadt., M., Haden, P., Hamer, J., H., & M., Lister, R., Petre, M., Sutton, K., Tolhurst, D., & Tutty, J. (2006). Predictors of Success in a First Programming Course. *In Proceedings of the 8th Australasian Conference on Computing Education*, *52*, 189–196.

Singer, Wing Hang Li , David R. White, J. (2013). JVM-Hosted Languages: They Talk the Talk, but do they Walk the Walk? *In Proceedings of the 2013 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools,ACM*, 101–112.

Sivasakthi, M., & Rajendran, R. (2011a). Learning difficulties of " object-oriented programming paradigm using Java ": students ' perspective. *Indian Journal of Science and Technology*, *4*(8), 983–985.

Sivasakthi, M., & Rajendran, R. (2011b). Learning difficulties of " object-oriented programming paradigm using Java ": students ' perspective. *Indian Journal of Science and Technology*, *4*(8), 983–985.

Smith, H. W. (1991). *Strategies of social research:* Holt, Rinehart and Winston.

Smith. (2012). *Research Methodology: A Step-by-step Guide for Beginners*.

Soloway, S. and. (1989). *Studying the novice programmer*. *H*illsdale, New Jersey, United States: Lawrence Erlbaum.

Souza, D. D., Hamilton, M., Thevathayan, C., Harland, J., Walker, C., & Muir, P. (2008). Transforming Learning of Programming: A Mentoring Project. *In Proceedings of the Tenth Conference on Australasian Computing Education*, *78*, 78–84.

Stroustrup, B. (1991). What is "Object-Oriented Programming"?(1991 revised version).

Tabachink, B. G. &Fidell, L. S. (2006). *Using Multivariate Statistics* (5th Ed.), USA: Pearson Education Inc.

Thompson, P. (1995). *Constructivism in education* (p. 159). Hillsdale, NJ: Lawrence Erlbaum.

Turner, J. A., & Zachary, J. L. (2001). Javiva: A Tool for Visualizing and Validating Student-Written Java Programs. *In ACM SIGCSE Bulleti*, *33*(1), 45–49.

Ulloa, M. (1980). Teaching and learning computer programming: a survey of student problems, teaching methods, and automated instructional tools. *ACM SIGCSE Bulletin*, *12*(2), 48–64.

Von Wangenheim, C. G., & Shull, F. (2009). voice of evidence. *IEEE*, *26*(2), 92–94.

VRajaraman. (1998). Programming Languages. *http://ezproxy.unimap.edu.my:Comparison and Classification of Programming Languages (Springer)*, 1–12.

Weisfeld, M. (2004). *The Object-Oriented Thought Process. (2nd ed.).Developer's Library*.

Weiss, M. A. (2000). Data structures and problem solving using Java. *ACM SIGACT News*, *29*(2), 42–49. doi:10.1145/288079.288084

Wenger, E. (1998). Communities of practice: Learning as A social practice. *The Systems Thinker*, *9*(5).

Wiedenbeck, S. (2005). Factors affecting the success of non-majors in learning to program. *Proceedings of the 2005 International Workshop on Computing Education Research - ICER '05*, 13–24. doi:10.1145/1089786.1089788

Wiedenbeck, S. et al. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers*, *11*, 255–282.

Wiedenbeck, S., & Labelle, D. (2004). Factors Affecting Course Outcomes in Introductory Programming, (April), 97–110.

Williams, K. C., & Williams, C. C. (2011). Five key ingredients for improving student motivation. *Research in Higher Education Journal*, 1–23.

Wilson, B. C., & Shrock, S. (2001). Contributing to Success in an Introductory Computer Science Course : A Study of Twelve Factors. *In ACM SIGCSE Bulletin*, *33*, 184–188.

Wit, K. De, Heerwegh, D., & Verhoeven, J. C. (2012). Do ICT Competences Support Educational Attainment at University? *Journal of Information Technology Education: Research 11. Available at http://www.jite.org/documents/Vol11/JITEv11p001-025DeWit1037.pdf [Accessed 22-05-2012]*, *11*.

Xinogalos, S., Sartatzemi, M., & Dagdilelis, V. (2006). Studying Students ' Difficulties In An Oop Course Based On Bluej. *In IASTED International Conference on Computers and Advanced Technology in Education*, 82–87.

Yau, J. Y., & Joy, M. (2004). Introducing Java: the Case for Fundamentals-first.

Zdancewic& Weirich. (2013). *Programming Languages and Techniques.*University of Pennsylvania, 1–387.

Zikmund, W., Babin, B., Carr, J., & Griffin, M. (2010). *Business research methods South*-Western Cengage: Canada.