

**UNIVERSITY STUDENTS' METACOGNITIVE
AND PROBLEM SOLVING SKILLS
TOWARDS
LEARNING A PROGRAMMING LANGUAGE**

**A THESIS SUBMITTED TO THE GRADUATE
SCHOOL OF APPLIED SCIENCES
OF
NEAR EAST UNIVERSITY**

**BY
RAJAA. JAWADI. H. FOURTI**

**In Partial Fulfilment of the Requirements for
the Degree of Master of Science
in
Computer Information Systems**

NICOSIA, 2017

RAJAA JAWADI. H

**THE EFFECT OF METACOGNITIVE SKILLS TOWARDS PROBLEM
SOLVING IN PROGRAMMING LEARNING**

**NEU
2017**

**UNIVERSITY STUDENTS' METACOGNITIVE AND
PROBLEM SOLVING SKILLS
TOWARDS
LEARNING A PROGRAMMING LANGUAGE**

**A THESIS SUBMITTED TO THE GRADUATE
SCHOOL OF APPLIED SCIENCES
OF
NEAR EAST UNIVERSITY**

**BY
RAJAA JAWADI.H.FOURTI**

**In Partial Fulfilment of the Requirements for
The Degree of Master of Science
in
Computer Information Systems**

NICOSIA, 2017

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Rajaa Jawadi Fourti

Signature:

Date:

ACKNOWLEDGEMENTS

My deepest gratitude goes to Prof. Dr. Nadire Çavuş, for her constant encouragement and guidance. She has walked me through all the stages of the writing of my thesis. Without her consistent and illuminating instructions, this thesis could not have reached its present form.

I would like to thank Prof. Dr. Dogan Ibrahim and Assist. Prof. Dr. Seren Başaran who have been very helpful through the duration of my thesis, this thesis could not have been achieved without their generous and professional assistance. Also, I want to thank to all jury members for their valuable comments.

This thesis is dedicated to my beloved family with unlimited thanks and heartfelt love, for they have believed in me and have sustained me throughout my life. A special feeling of gratitude to my hero, my brother Faraj Aljwadi who is indeed my inspiration and the man who led me into the treasures of knowledge. I would like to thank my husband for his unlimited and unconditional love, and to my parents who support me in everything all the time. I would like to thank my mother for her unlimited and unconditional love, and to my father who taught me how to be a real man before everything, and taught me that knowledge must be learned for its own sake. I would like to thank my sisters and brothers for their encouragement and constant love they gave me.

Eventually, to that long list of friends who have supported me all over the way from the early stage of my study until the last word of this thesis, thank you all for all the love and help you gave me, I couldn't be here without your existence in my life, this thesis would not have been possible.

To my parents...

ABSTRACT

The complexity of the educational processes made it necessary to look for new appropriate strategies. It is important that students acquire metacognitive skills so that they can keep abreast of the great progress in Science and Technology. Educational methods should be questioned about their role in preparing a student to possess not only thinking but being able to have metacognitive skills. Computer programming skills require a good quality of education through a comprehensive development of the educational processes and the knowledge of the latest approaches which require the preparation of students' abilities to choose the most appropriate strategies. Metacognitive skills increase students' abilities to understand how to think positively, and how to solve the problems they face in Computer Science. The Thesis attempts to provide an understanding of the metacognitive skills towards problem solving in programming language learning amongst university students in North Cyprus. Research based model and questionnaire were used in the study where data were collected randomly from 300 volunteered students. The students were chosen from departments of Computer Information Systems, Computer Engineering, Information Technology and Management Information Systems during 2016-2017 Fall semester. The dependent variable in the study is metacognitive skills for computer programming, and the independent variable is problem solving skills. SPSS was used to analyze the data, the descriptive statistics were used to analyze the characteristics of the study sample and the estimates of their responses. Pearson's correlation was used to study the effect of use metacognitive skills towards problem solving skills that students face when learning programming languages. After statistical analysis of the collected data, there was a negative correlation between metacognitive skills and problem solving skills. This study also showed a strong point among students in the skills of planning and monitoring, and a weakness among students in the skills of regulation.

Keywords: Programming learning; programming; metacognitive skills; problem solving; problem solving skills

ÖZET

Eğitim süreçlerindeki karmaşa, uygun yeni stratejiler aramayı zorunlu kılmıştır. Öğrencilerin Bilim ve Teknoloji alanındaki büyük ilerlemelere ayak uydurabilmeleri için bilişsel becerileri kazanmaları büyük önem arz etmektedir. Bu nedenle, eğitim metodları öğrencileri sadece düşünmeye değil, üstbiliş becerileri geliştirmeye hazırlamalıdır. Bilgisayar programlama becerileri, öğrencilerin en yeni ve en uygun stratejiler seçebilme becerilerini kazanmalarını sağlayan iyi bir eğitim gerektirir. Üstbiliş becerileri, öğrencilerin pozitif düşünme kabiliyetlerini artırır ve bilgisayar bilimindeki konularda problem çözebilme becerilerini geliştirir. Tezde, Kuzey Kıbrıs'ta üniversite öğrencilerinin bilgisayar programlama dili öğreniminde üstbiliş becerilerinin problem çözme becerilerindeki önemi araştırılmıştır. Araştırma tabanlı model ve anket kullanılarak 300 gönüllü öğrenciden veri toplanmıştır. Ankete katılanlar, Kuzey Kıbrıs'ta 2016-2017 Sonbahar döneminde Bilgisayar Enformatik Sistemleri, Bilgisayar Mühendisliği, Bilişim Teknolojileri ve Yönetim Bilişim Sistemleri bölümlerinde eğitim gören öğrencilerden seçilmiştir. Çalışmada bağımlı değişken olarak bilgisayar programlamada üstbiliş becerileri alınmıştır. Bağımsız değişken olarak ise problem çözme becerileri alınmıştır. Toplanan verilerin analiz ve yorumlanmasında SPSS programı kullanılmıştır. Üstbiliş becerilerinin programlama dili öğrenirken problem çözmedeki katkılarını öğrenmek için Pearson Korelasyonu kullanılmıştır. Toplanan veriler istatistiksel olarak analiz edildikten sonra üstbiliş ve problem çözme becerileri arasında negatif korelasyon olduğu belirlenmiştir. Çalışmada ayrıca öğrencilerin planlama ve izleme becerileri arasında yüksek korelasyon olduğu ve düzenleme becerilerinde düşük korelasyon olduğu ortaya çıkmıştır.

Anahtar kelimeler: Programlama öğrenme; programlama; üstbiliş becerileri; problem çözme; problem çözme becerileri

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	i
ABSTRACT	iii
ÖZET	iv
TABLE OF CONTENTS	v
LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF ABBREVIATION	x
CHAPTER 1: INTRODUCTION	1
1.1 Overview.....	1
1.2 The Problem.....	4
1.3 The Aim of the Study.....	4
1.4 Importance of the Study	5
1.5 Limitations of the Study.....	5
1.6 Overview of the Thesis.....	6
CHAPTER 2: RELATED RESEARCH	7
2.1 Research on Difficulties in Programming	7
2.2 Effects of Metacognitive Skills on Problem Solving.....	8
CHAPTER 3: THEORETICAL FRAMEWORK	10
3.1 Computer Programs.....	10
3.2 Computer Programming.....	11
3.3 Importance of Learning Programming.....	12
3.3.1 Advantages of object-oriented programming.....	12
3.3.2 Disadvantages of object-oriented programming.....	13
3.4 Programming Difficulites	13

3.5 Programming Language Features.....	15
3.6 Educational Programming Environments.....	16
3.7 Programming Learning Strategies.....	19
3.8 Problem Solving Approach.....	24
3.8.1 Steps in problem solving.....	24
3.8.2 Problem solving strategies for programmers.....	25
3.9 Metacognitive Skills	26
3.9.1 Metacognitive skills for programmer.....	27
3.9.2 Metacognitive steps for problem solving	29
3.9.3 Metacognitive components for problem solving.....	31
CHAPTER 4: RESEARCH METHODOLOGY.....	35
4.1 Research Model.....	35
4.2 Participants.....	37
4.3 Data Collection Tools.....	39
4.4 Data Analysis	41
4.5 Procedure	42
4.5.1 Ethical considerations.....	43
4.6 Research Schedule	44
CHAPTER 5: CONCLUSION AND RECOMMENDATIONS.....	46
5.1 Know Programming Language of Students	46
5.2 Difficult of Stuentns on Learning Programming Language.....	47
5.3 Metacognitive Skills of Students for Computer Programming	50
5.3.1 Planning	51
5.3.2 Monitoring	52
5.3.3 Regulation	53
5.4 Problem Solving Skills of Students for Computer Programming.....	54
5.4.1 System	56
5.4.2 Design	56
5.4.3 Coding	57

5.4.4 Testing	58
5.5 Relationship between Sub-Dimensions of Metacognitive and Problem Solving Skills..	59
5.6 Relationship between Metacognitive Skills and Problem Solving Skills.....	62
CHAPTER 6: CONCLUSION AND RECOMMENDATIONS.....	65
6.1 Conclusion.....	65
6.2. Recommendations.....	66
REFERENCE.....	67
APPENDICES.....	75
Appendix A: Rapporteur of the Scientific Research Ethics Committee.....	76
Appendix B: The Questionnaire	77
Appendix C: Similarity Report.....	81

LIST OF TABLES

Table 3.1	Showing different metacognitive components and implications in problem solving	33
Table 4.1	Important about chosen universities	37
Table 4.2	Important demographic data of participants.....	38
Table 4.3	Problem solving skills.....	41
Table 4.4	Score boundaries of the 5 Likert scale of level of knowledge of programming language.....	41
Table 4.5	Score boundaries of the 5 Likert scale of level of difficulties faced by student.....	42
Table 4.6	Score boundaries of the 5 Likert scale of level of problem solving skills	42
Table 5.1	Mean and standard deviation programming language that the students know.....	47
Table 5.2	Mean and standard deviation for each items of difficult do students in programming language are to learn.....	49
Table 5.3	Mean and standard deviation for metacognitive skills of students for computer programming questionnaire.....	51
Table 5.4	Mean and standard deviation for every item of the planning dimension	52
Table 5.5	Mean and standard deviation for every item of monitoring dimension..	53
Table 5.6	Mean and standard deviation for every item of regulation dimension...	54
Table 5.7	Mean and standard deviation for problem solving skills of students for computer programming of the questionnaire.....	55
Table 5.8	Mean and standard deviation for each item of system analysis dimension	56
Table 5.9	Mean and standard deviation for every item of design dimension.....	57
Table 5.10	Mean and standard deviation for every item of coding dimension.....	58
Table 5.11	Mean and standard deviation for every item of testing dimension.....	59
Table 5.12	Relationship between sub-dimensions of metacognitive skills and problem solving skills.....	62
Table 5.13	Relationship between metacognitive skills and problem solving skills..	63

LIST OF FIGURES

Figure 3.1	Object-oriented programming concepts.....	15
Figure 3.2	Programming environment for Object Karel.....	17
Figure 3.3	Screenshoot of Raptor visul programming environment	18
Figure 3.4	An example of BlueJ project people.....	20
Figure 3.5	Shows a screenshot of Alice development environment.....	22
Figure 3.6	Screenshot of a source code for calculating compound interest.....	24
Figure 3.7	Stages in metacognitive skills for solving programming problems.....	32
Figure 3.8	A model of metacognitive skills and how to effectively solve problems.....	34
Figure 4.1	The effect of metacognitive skills towards problem solving in programming learning.....	36
Figure 4.2	Structure of the questionnaire.....	39
Figure 4.3	Research procedure.....	43
Figure 4.4	Research schedule of the study.....	44
Figure 4.5	Gantt chart for the thesis.....	45
Figure 5.1	Scatter Plot between planning dimension and problem solving skills	60
Figure 5.2	Scatter Plot between monitoring dimensions and problem solving skills...	60
Figure 5.3	Scatter Plot between regulation dimension and problem solving skills.....	61
Figure 5.4	Scatter Plot for relationship between metacognitive skills and problem solving skills.....	63

LIST OF ABBREVIATIONS

ANNOVA:	Analysis of Variance
MS:	Metacognitive skills
PS:	Problem solving
OO:	Object-oriented
CIS:	Computer information system
IT	Information technology
CE	Computer engineering
MIS	Computer information system
NEU:	Near east university
CIU:	Cyprus international university
EMU:	Eastern mediterranean university
EUL:	European university of lefke
M	Mean
SD:	Standard deviation

CHAPTER1

INTRODUCTION

This chapter describes the problem, aim of the research, importance of the research, limitations of the research and overview of the Thesis.

1.1 Overview

Programming language is an extremely helpful skill and maybe a rewarding profession, it has recently become the interest for software engineers and understudies enthusiasm for programming have developed quickly, and basic programming courses are getting well known unlike before (Wang & Hwang, 2017)

Myers et al. (2016) conducted a research to find out the problems faced by students in programming. The results of the study showed that changing the design was more hard to comprehend than testing the variables and refreshing them. Strikingly, scientists discovered that novices thought that it was simple and easy to compose recursive capacities subsequent to finding out about iterative capacities. Other researchers, Luxton-Reilly and Petersen (2017) discovered that pointers were positioned at the highest difficulty in addition to expressions and language differences, particularly in C dialect. Also, the scientists additionally found that polymorphism, the giving was additionally arranged as difficulty. The one of most complex human conduct is the way toward taking care of the problem solving, which requires aptitudes and high subjective abilities to control the issue and find proper answers for it. The problem-solving process located at the top of the hierarchy of education (Mahmoud, 2013).

Rum and Ismail (2014) computer programming language is a difficult and intricate operation for many students which places a heavy burden on students. Studying programming courses is the main defy, for students, especially novices because of the difficulties in learning programming skills that are out their capacities (Apiola et al., 2011). Learning computer programming is not a difficult process because of basic concepts but the most difficult aspects lie in planning to write the program (Fu et al., 2017).

The main purpose of solving software problems is to enable students to think and develop their skills in front of the problem and to judge these skills and their impact on flexibility

and refinement of thinking (Avargil et al., 2018). Although, metacognition skills contain how an individual can investigate thoughts, locate the perfect strategy for access to a suitable way and how to apply them to the learning process. To take care of the issue, the students must acknowledge and comprehend their capacities and how to perform subjective aptitudes, for example, learning and problem solving skills (Özsoy & Ataman, 2017). And then point students to efficient strategies for the computer programming process (Kasemsap, 2017). The former studies carried out on metacognition and problem-solving studies have shown that metacognitive education boost students capacity to resolve problems best because metacognitive strategies develop students ability to solve difficulties that they faced when learning programming (Mokos & Kafoussi, 2013). The problem-solving skills process don't demand just that the students know about the helpful knowledge components, additionally, should be able to arrange and control these cognitive components any assignment to problem solving (Wang & Hwang, 2017). In this, Sarver (2006) points out that the knowledge skills include planning, monitoring, and evaluation through which the learner can control his knowledge of the good method through improving his ability in solving the problem, and the metacognitive skills allow self-learning as it helps students to Self-perception of their thinking.

The main purpose of educational establishment is teaching students how to make use of processes such as planning, how to apply the knowledge, how to monitor, how to regulate and reflect (Azevedo, 2009). Metacognition indicates to higher thinking in disposition including the active control of the knowledge process involved in learning. Activities such as planning how to handle a particular learning task, monitoring the understanding, and evaluating progress towards completing the task are beyond the cognitive nature (Flavell, 1979). Because knowledge skills is a key factor in successful education, it is important to identify the activity that goes beyond the cognitive process to reach the ideal way to improve students thinking through cognitive control (Avargil & Lavi, 2018). Therefore, Ali et al. (2016) stated that metacognitive thinking makes individual think in a state of continuous research and investigation and conscious observation that help them to better deal with situations and problems.

Tas et al. (2014) defined that metacognition is a composite activity of skills that refers to knowledge control, and planning for learning through the choice of appropriate strategies,

and monitoring processes, and to evaluate these processes. Other researchers, (Costa and Kallick, 2001) a pyramid model was introduced to explain the skills of metacognition where they reached a conclusion that metacognition is a composite activity of skills that refers to knowing metacognition skills and control it, and planning for learning by selecting appropriate strategies, monitoring knowledge processes, and evaluating these processes. On the other hand, Dawson et al. (2008) defined that metacognition as a process that co-ordinates data, practices, goals, and plans. Metacognition, which basically means thinking, generally includes various skills that are related to acquiring knowledge and thinking, which are critical thinking, contemplative thinking, problem solving (PS), and making decisions. People, who have skills beyond the most advanced knowledge, are also the best problem solvers, decision-makers, and critical thinking of other people in general. In fact, the more complex the programming problem, the greater the need for metacognitive knowledge, meaningful reflection and positive reactions (Havenga, 2011). Montague et al. (2014) pointed out that beyond knowledge means thinking about a thinking process, and therefore it is due to a high mental capacity that interferes with the process of learning in terms of finding a learning plan, using appropriate skills and strategies to solve problems, studies show that meta-knowledge skills are important in predicting the academic achievement of learners, it helps them effectively distinguish between information they know and do not know.

Kafadar (2012) Stated that the definition of a problem solving process involves the individual's use of all of his practical and cognitive skills, including cognitive activities such as planning, evaluation, and monitoring. Solving the problem requires three main conditions: Firstly think about the problem and guide the behavior towards the goal, then find a law or strategy that can help achieve the desired goal and finally treat these laws or strategies by putting them in action. At this stage, it is necessary to identify and define the sub-objectives according to the type of problem, then solve the problem and reach the objective. Furthermore, Ramesh and Anandaraj (2014) have found out that there is a major correlation between students' metacognition and problem solving skills. Nevertheless, problem solving requires the individual to use all his cognitive and reflective abilities, which involve the skills of knowledge such as planning, organization, and exploitation in the direction required to overcome the problem (Kafadar, 2012). Problems occur when there is a gap between the individual and the goal to be achieved and is also a complex

process in our lives (Baars et al., 2017). Therefore, problem solving is also a complex process that exists in everyday practice (Ali et al., 2016). Therefore, solving the problem is not only in the field of computer programming but it exists in all areas that affect the daily experiences and therefore can be defined as knowledge and mental processes which can be directed to arrive at the appropriate strategy to solve the problem (Marjorie et al., 2014)

In this thesis, we have focused on the effect of metacognitive skills towards problem solving in programming learning, and the strategies to use in problem solving and improve performance towards thinking and regulation. Also, enable all students to have strategic thinking and examine the relationship between metacognitive knowledge and performance on the task to problem solving.

1.2 The Problem

In the literature, many researchers (Siswati & Corebima, 2017; Anandaraj & Ramesh, 2014; Safari & Meskini, 2015) have pointed out that metacognition is important when dealing with problem solving. Arslan et al. (2013) conducted a research in Turkey to find the important factors that lead students to use metacognitive skills in computer programming. Despite the proliferation of computer programming, students are still encountering difficulties in learning programming languages. Moström (2011) hoped to reveal the right solutions that will help students improve their skills on enhancing creative thoughts and problem solving within a metacognitive framework and overcoming difficulties. Furthermore, students have the problem to learn programming languages and they think it is difficult but learn programming is useful and requirements for real life. Therefore, suggestions to improve teaching methods will be helpful for universities developing in countries especially in North Cyprus.

1.3 Aim of the study

The foremost aim of this study is to find out about the effect of metacognitive skills on problem-solving skills in programming among university students. To achieve the main aim, we examine the following research questions:

1. Which programming language do students know?
2. How difficult do students believe each of the topics in a programming language are to learn?

3. What are metacognitive skills of students for computer programming?
4. What is problem solving skills of students for computer programming?
5. Are there any relationships between sub dimensions of metacognitive and problem solving skills among students?
6. Is there any relationship between metacognitive and is problem solving skills of students?

1.4 Importance of the Study

Solving problems has always been a challenge as well as teaching students programming languages. This study is important to educational institutions mainly targeting computer science and other technical disciplines which offer programming courses. Solving problems has always been a challenge for teachers teaching students programming languages and knowing how to apply their knowledge to improve their programming skills and enhancing critical thinking. One of the biggest problems for programming novices is that there is a huge gap between the intuitive way in which they think and the way of thinking that has to be suitable for computers. Most students do not have enough level of software knowledge and they have difficulties in understanding of programming tasks and in designing of appropriate algorithms most students start to learn to programme in single context before learning structure and style (Churchill et al., 2013). Moreover, teachers don't use new technologies or new methods of teaching (Hu, 2004). Most efforts are aimed at bringing the user closer to the system and increase students' performance and learn to programme by use new way of thinking to enable them to use a programming language in order to solve problems. Therefore, the metacognition is important and proper process for solving the problem. In this study, a survey will be carried out to discuss the impact of metacognitive education on problem solving in learning programming languages. Findings from the study will be beneficial to researchers who are interested in the same area of study as well as authors who are keen to address the challenges students face in learning programming.

1.5 Limitations of the Study

There are quite a number of limitations that have been noted in this study. These limitations should be considered for future research. The research was conducted over a short period of time during the spring semester, a longitudinal research should be

considered in future that is done over a longer period of time. The research only focus on four universities in Northern Cyprus namely Near East University (NEU), Cyprus International University (CIU), European University of Lefka (EUL), and Eastern Mediterranean University (EMU). The study consists of 300 students from four universities in North Cyprus and because the study period was not enough to collect data from a larger community. A larger target group should be considered for future research to yield generalized results of the difficulties in learning programming language in North Cyprus as a whole

1.6 Overview of the Thesis

This study is divided into six chapters as follows:

Chapter 1: This chapter is the first chapter of the thesis and this chapter includes the problem statement, aim of the study, and the importance of the study, research questions and an overview of other chapters to follow.

Chapter 2: It discusses previous research that has been done by other researchers interested in understanding the effect of metacognitive skills towards problem solving in programming learning. This chapter explains research student's difficulties, effects of metacognitive skills on problem solving.

Chapter 3: This chapter is designed as a medium to help institutions improve the delivery of programming courses. The chapter explains the theoretical framework of the topic under research.

Chapter 4: Research model, participants, sample selection, data collection and tools, data analysis and the research procedure were explained in detail in this chapter. Reliability tests are also shown for the survey tool used.

Chapter 5: The chapter reveals the results found. Results are then discussed with respect to the fundamental objectives of the research

Chapter 6: It is about the conclusion of the entire research study and recommendations of the thesis, suggestions, and for future studies

CHAPTER 2

RELATED RESEARCH

2.1 Research on Difficulties in Programming

Bouvier et al. (2016) emphasized that learning programming skills is generally considered hard, and programming courses often have high dropout rates. That it takes about 10 years for a novice to become an expert programmers. Kirsti (2004) pointed out that students find it difficult to understand program execution since it takes a long time for a beginner to understand the mechanism and the operation taking place behind the code. Students indicated that they find it difficult to understand that the instructions being executed are the ones created in the previous instructions. Milne and Rowe (2003) also indicated that students found virtual functions as moderately difficult contrary to what teachers perceived. Teachers indicated that it was one of the main challenges students face.

Madden and Chambers (2002) found out that Java was easier to learn for novices compared to C language due to the absence of pointers in Java however, the researchers did not phase out the assumption that object oriented programming is difficult. In addition to these findings, the researchers conducted a research at Anna University for undergraduate students of Engineering and Technology who were doing Java courses. Students indicated that they prefer to learn through assignments and tutorials rather than class lectures. In the survey, 195 students participated. Findings showed that 67% of the students have a personal computer to practice with. Among the ten listed programming topics, students were asked to rank the topics in their level of difficulty. Starting from the most difficult topic to the least difficult topic, the order was concurrent programming, UI components with swings, generic programming, exceptions and assertions, event handling, interfaces and inner classes, graphics programming, object oriented access controls, object orientation and fundamental programming structure in Java.

Kirsti (2004) conducted a study in Bulgaria to find out challenges that teachers face when teaching programming. The majority of the respondents indicated that students who are taught procedural programming first find it difficult to switch and understand how object oriented programming work. Kunkle (2010) suggested that universities and other academic institutions should consider teaching students the principles of object oriented

programming first as an introductory course before they are taught procedural programming. Studies have shown that student's master object oriented programming faster when it is introduced first (Kirsti, 2004). Maheswari et al. (2017) highlighted that apart from introducing object oriented programming, it is also crucial for students and teachers to understand that programming is for scholars, it requires abstract thinking.

Milne and Rowe (2002) conducted a study to find out the difficulties of C++ programming for teachers and students. Findings revealed that students rated having less difficulties compared to the rating that teachers indicated. Analysis of these ratings showed that the reason lies in that students tend to believe in themselves that they have understood yet on the other hand, teachers see the remaining flaws that in programming coursework and examinations.

2.3 Effects of Metacognitive Skills on Problem Solving

Safari and Meskini (2015) conducted a research in Iran to find out the effect of metacognition instruction on problem solving for students doing health sciences. The sample size constituted of 40 students in their second semester who were enrolled at Kermanshah University of Medical Sciences. Findings revealed that there was a significant effect between metacognitive instruction and problem solving skills. Similar findings were also found by Ramesh (2014) who also found out that there was a significant correlation between metacognition and problem solving. In addition, Mokos and Kafoussi (2013) also pointed out that metacognitive approach to learning helps students improve their problem solving skills. Harandi et al. (2013) also found out that metacognitive skills have a significant effect on problem solving skills, however, the researchers did not find any difference between orientation-avoidance and personal control components.

In the literature, many researchers (Aurah et al., 2014; Hong et al., 2017) also conducted studies to find out the effect of metacognitive skills on improving problem solving skills among university students. They all found out that in the control group, the effect of conventional teaching was evident on problem solving. In addition, they also found out that orientation-avoidance and personal control components are part of problem solving skills. There was not much difference between the experimental group and the control group, however emphasis was put by all the researchers that problem solving confidence in the control group was greatly affected by conventional teaching methods.

Annemieke et al. (2012) conducted a study to find an efficient measurement of metacognition in mathematical problem solving. The sample constituted of 42 students who were randomly selected from a grade 5 class. The study involved a series of steps which were measured. Firstly the students were given a word problem and had to rate their confidence in tackling the problem first. Secondly, the students had to make a sketch map which would help them in solving the problem. Thirdly, the problem had to be solved and lastly the student was required to rate his/her performance before the results were shown. Results showed that metacognitive was relatively low on all measures which show that metacognitive skills are still in the early development stage among elementary students.

In addition, the literature, many researchers (Arslan, 2014; Arslan & Akin, 2014; Arslan et al., 2013) pointed out that important factors that constitute to academic success are metacognitive and self-regulation. Arslan (2014) conducted a research in Turkey to examine the relationship between metacognition and self-regulation in web based teaching environments. Findings revealed that metacognitive skills in web learning environments had a positive effect on problem solving skills and overall academic success. In conclusion, the researcher pointed out that students with high levels of self-regulation had higher levels of metacognition.

Anandaraj and Ramesh (2014) conducted a study in India to find the relationship that exists between metacognition and problem solving among students studying physics. Findings revealed that the level of metacognition among students varied between gender and college. Furthermore, results also showed that there was a significant difference between students studying physics with regard to gender. In addition to that, results also showed that female students were better than male students in their level of metacognition and urban students had higher metacognition levels compared to rural students. In conclusion, the findings showed that there was a significant relationship between the metacognition levels of students studying physics and their problem solving ability.

CHAPTER 3

THEORETICAL FRAMEWORK

This chapter explains the theoretical construct of this research. The chapter explains computer programs, computer programming, importance of learning programming, the programming difficulties, Programming Language Features, Educational Programming Environments, Programming Learning Strategies, Problem Solving Approach, Metacognitive skills.

3.1 Computer Programs

Biju (2013) defined a computer program as a list of instructions that are responsible for telling a computer what to do. In addition, the researcher stated that all processes done by computers and other electronic devices are controlled by computer programs which are written in different programming languages. Computer programs are used in our everyday life to run and operate different devices as well as view web pages. Notable examples of everyday use of computer programs include web browsers such as Google Chrome, Mozilla Firefox, spreadsheet documents and video games. Castro et al. (2016) described how computer programs function. The researchers, describes a computer program as a file that resides in the computer's hard drive. When users run applications on digital devices, the file is read and processors located in the device interprets the instructions and executes the desired output. Computer programs are written by programmers using programming languages such as C++, Java, Python and many more languages available in the market. In order for computers to understand the code written, programmers make use of compilers which are responsible for translating the code into machine readable format which is in binary form. Furthermore, the researchers pointed out that it is crucial for researchers and users to note that there are also bad computer programs that are developed illegally by users who plan to destruct the functioning of computer systems which are known as malware and spyware used to steal important user information.

Computer programs have become crucial in every business industry ranging from education, finance, manufacturing, agriculture and many other sectors. It is important for users to understand that computer programs are not only used in programming classes by programmers but they form a part of our everyday life. Crowfoot (2012) focused on explaining the four main computer programs that are used in everyday life by users in

different sectors. The 4 common computer programs and their applications are explained in detail below:

- **Web browsers:** This is the main computer program used every day. Examples include Google Chrome, Mozilla Firefox, Internet Explorer and other web browsers available. Millions of users access Facebook, Google, Twitter etc. every day, without web browser computer programs, users would not be able to access the different websites they access on a daily basis.
- **Microsoft Office and Outlook:** In order to solve every day problems users make use of computer programs such as Microsoft word, PowerPoint, Excel and outlook for communication purposes. Spreadsheets are used to save data in different work places.
- **Antivirus:** Anti-virus are computer programs that are developed to protect computers and other devices from viruses that are available on the web. These computer programs operate in the background, preventing malicious software from entering the computer.
- **PDF Readers:** These computer applications include Adobe Reader and many others and are the most popular and preferred format of distributing files on email platforms. These computer programs are embedded with security features that check viruses in documents to be sent via the web.

3.2 Computer Programming

Computer programming is the evolving process involving the implementation of various sets of instructions to permit a computer to do a certain task by means of programming language to solve problem. Özgür (2017) explained the classification of programming languages as described in the section below:

- **Web languages:** Used in creating as well as editing pages on the web which involves putting plain text on webpages, to access as well as retrieving data from a database. Web languages include HTML XML, JavaScript, VBScript, PHP, Java, and ASP.
- **Procedure-oriented:** Style of programming in which single operations are used in a program and are grouped in logical units which are called procedures or functions

- ***Object oriented programming:*** Style of programming involves combining or encapsulating data into a solo program entity called object. Examples of these languages include C, C++, Java and PHP. The programs organize data and is collected using a group of interacting objects, each has its very own data space and functions. It is possible to make objects reusable through encapsulating the whole thing needed to operate.
- ***Software development:*** Languages are used for creating executable programs. In addition, these languages are capable of creating simple console programs that are capable of printing text to the screen. It varies greatly in terms of power and complexity. Such as C, C++, C#, Pascal, Delphi, Visual Basic.

Object-oriented programming languages are the most used programming languages. The main advantage for Object-oriented programming is that it is easy to maintain as well as modifying existing codes (Henry, 2013). Development time is cut considerably and this makes adjustments of program much simpler.

3.3 Importance of Learning Programming

In the literature many researchers (Todorova & Donchev, 2004; Kirsti, 2008; Kunkle, 2010) have stated the importance of object-oriented programming and Kunkle (2010) went on further to explain the advantage and disadvantages of object-oriented programming as follows:

3.3.1 Advantages of object-oriented programming:

In the literature, many researchers (Henry, 2013; Butler & Morgan, 2001) explained the various benefits that come with using object-oriented programming language. The main advantages are described below:

- ***Improved software-development productivity:*** Object-oriented (OO) programming is extensible and modular therefore, it provides separation of duties in programs and new attributes and behaviours can be added. Compared to procedural programming, with object-oriented programming software development is improved due to extensibility, reusability and modularity factors that object-oriented programs exhibit.

- **Improved software maintainability:** Due to modular properties that object-oriented programs exhibit, software is easier to maintain and updates which can be done easier without making large-scale changes.
- **Faster development:** Reusability properties exhibited in object-oriented programs enable faster development through the use of rich libraries of code embedded in the programs.
- **Lower cost of development:** The overall cost of software development is lowered since software can be reused therefore more effort is diverted to object-orientation analysis and design.
- **Higher quality software:** It has been reported that object-oriented programming results in high quality software contrary to using procedural languages (Donchev & Todorova, 2008).

3.3.2 Disadvantages of object-oriented programming

Although Object Oriented (OO) Programming has been seen as a favourable programming language by many researcher and developers who have created good systems in the past, the same language has pitfalls which are explained bellow:

- **Steep learning curve:** Due to the nature of object-oriented programming, many programmers find it challenging especially in understanding key programming concepts such as polymorphism and inheritance. The interaction of objects makes it complex to create programs.
- **Larger program size:** Programming using object-oriented languages involves more lines of code which is contrary to procedural languages.
- **Slower programs:** Due to several lines of code instructions are executed in object-oriented programming therefore, it makes the programs slower compared to procedural based programs.

3.4 Programming Difficulties

Butler and Morgan (2007) explained that enhancing the programming teaching industry is crucial to identify and understand the challenges that learners face in programming courses. In addition, it is also important to identify the challenges that teachers encounter in teaching programming courses. Gomes and Mendes (2007) explained some difficulties to learning programming language in following points:

- ***Teaching is not personalized:*** Personalized supervision is more ideal where there is a teacher. Direct feedback explaining the problems which students find difficult could help. However, it may be straining to give such feedback.
- ***Teacher's strategies doesn't apply to all students learning styles:*** Individuals tend to approach new materials in different ways as they learn in several ways. Programming techniques tend to differ from student to student as learning styles are different and numerous ways can be used to arrive at the same conclusion or solve the same problem. The learning style differ from students as students are been taught through several means and tend to have different approach when it comes to learning new theory
- ***The teaching of dynamic theory with the aid of static materials:*** Programming consists some dynamic perception which are thought with the aid static objects such as (projected presentations, verbal explanations, tables, texts, and so on).
- ***Students use inaccurate study technique:*** Most students prefer to answering equations from memorized formulas, without a thorough understanding of many concepts as well as mastering many study options for programming.
- ***Most Students don't work hard enough to comprehend programming proficiency:*** Intense work is required for programming languages, solo study and textbook alone may not be sufficient.
- ***Most students lack knowledge in tackling programming:*** Without knowledge of solving problems, it is difficult for students to create algorithms. Finding solutions to problems requires adequate knowledge of programming which most student's lack.
- ***Many Students lack programming knowledge:*** It has been observed that many students face difficulty in programming due to misconceptions.
- ***Students lack mathematical and logical knowledge:*** The surrounding causes much objections which includes tacit or distinct mathematical concepts, particularly the theory which are essential for regular programming problems.
- ***Students lack insufficient programming knowlegde:*** Apparently, in most cases there is a blank space missing inbetween generic problem solving and programming problem solving. So therefore, it is important that the environment aids programmer to make this transition phase much lighter.

- **Programming demands a high level of abstraction:** The student's abstraction capacity must be developed.
- **Programming languages have very heterogenous structure:** The context lessen aspects inherent to language structure, stressing the algorithmic and problem solving progress. Students are more likely to concentrate basically in solutions which do not involve complex syntaxes.

3.5 Programming Language Features

In the literature the researchers (Madden & Chambers, 2002) have described the structure of object-oriented programs as depicted in Figure 1 below:

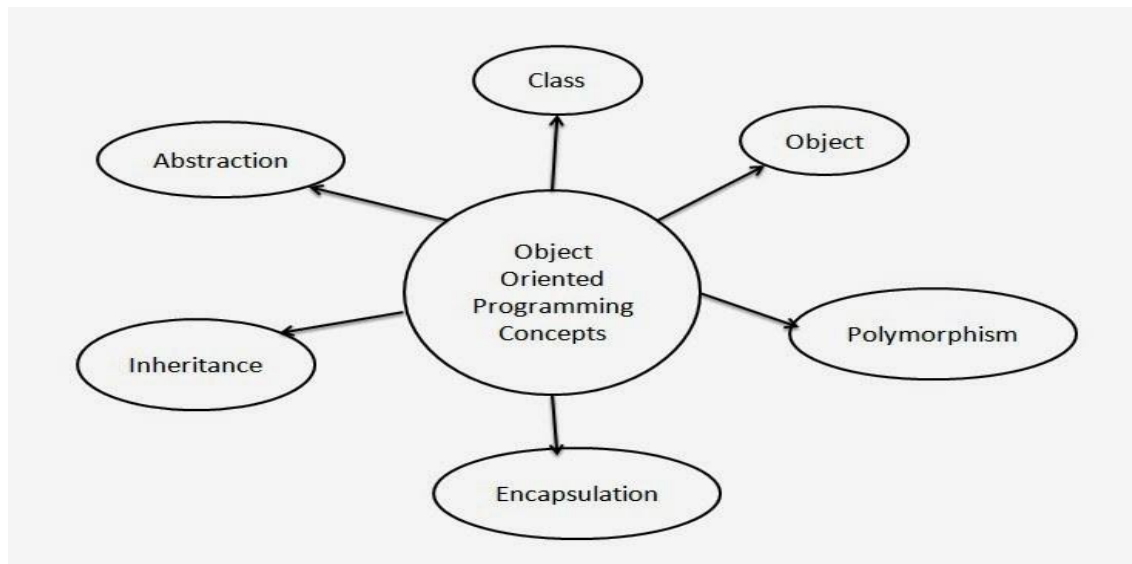


Figure 3.1: Object-oriented programming concepts (Madden & Chambers, 2002)

The structure of object-oriented programs is divided into six elements that Madden and Chambers (2002) explained below:

- **Objects:** An instance of a class is referred to as an object and form the basic run-time entity in an object-oriented system. Objects interact with each other by sending messages to each other during run-time as well as interaction without knowledge of underlying source code.
- **Classes:** This refers to group of objects of similar type and forms the basic entity of an object-oriented system. It is in this class that methods and variables of an object are defined.

- **Data Abstraction and Encapsulation:** The act of representing essential features is called abstraction and it does not include explanations and background details. Classes are referred to as a list of abstract attributes and they also use the some concept of abstraction. Encapsulation refers to the process of storing data and functions in a single unit. Data can only be accessed by functions which are stored in a class.
- **Inheritance:** This refers to the process whereby properties that objects of other classes are exhibit are acquired. Re-usability in inheritance enables additional features to be added to existing classes without any modification of the classes. Anew class is derived from an existing class and the output will be a new class that has features of the combined classes.
- **Polymorphism:** This refers to a programs ability to exhibit more than one form. In different instances, an operation may have different behaviours based on the data type used in the operation. Polymorphism is mainly used in implementing inheritance.

3.6 Educational Programming Environments

Programming environments have been devised as a strategy to help students learn programming easier and help eliminate the negative mind set portrayed about programming. Jimenez-Peris et al. (2000) highlighted the basic requirements for any professional educational programming environment that are listed below:

- Programming environments must be up-to-date. Compilers, user interface and environment functionality must be up-to-date.
- Programming environments must provide higher working models that do not complicate explanations by simply mirroring the implementation.
- Programming environments must draw attention on understanding not merely assistance in development. When errors occur, explanations should be understandable.
- Programming environments must cater for non-academic activities which include group work, written and oral communication.

On the Internet there are much more environment for programming learning. Some of them explained as follows:

a. Object Karel Programming Environment

Xinogalos et al. (2006) described Object Karel as an educational programming environment based on a special object-oriented mini-language. The environment has several features which are important for analogous environments. The editor is structured in a way that supports program development more than syntactic details. In addition, the environment provides a step by step process of program execution as shown on Figure 3.2 below. Furthermore, the structure editor supports program development rather than a detailed focus on the programming language. It is designed in such a way that supports in understanding programming structures and flow of control. In addition, it has built-in tutorials to aid as a teaching and learning tool.

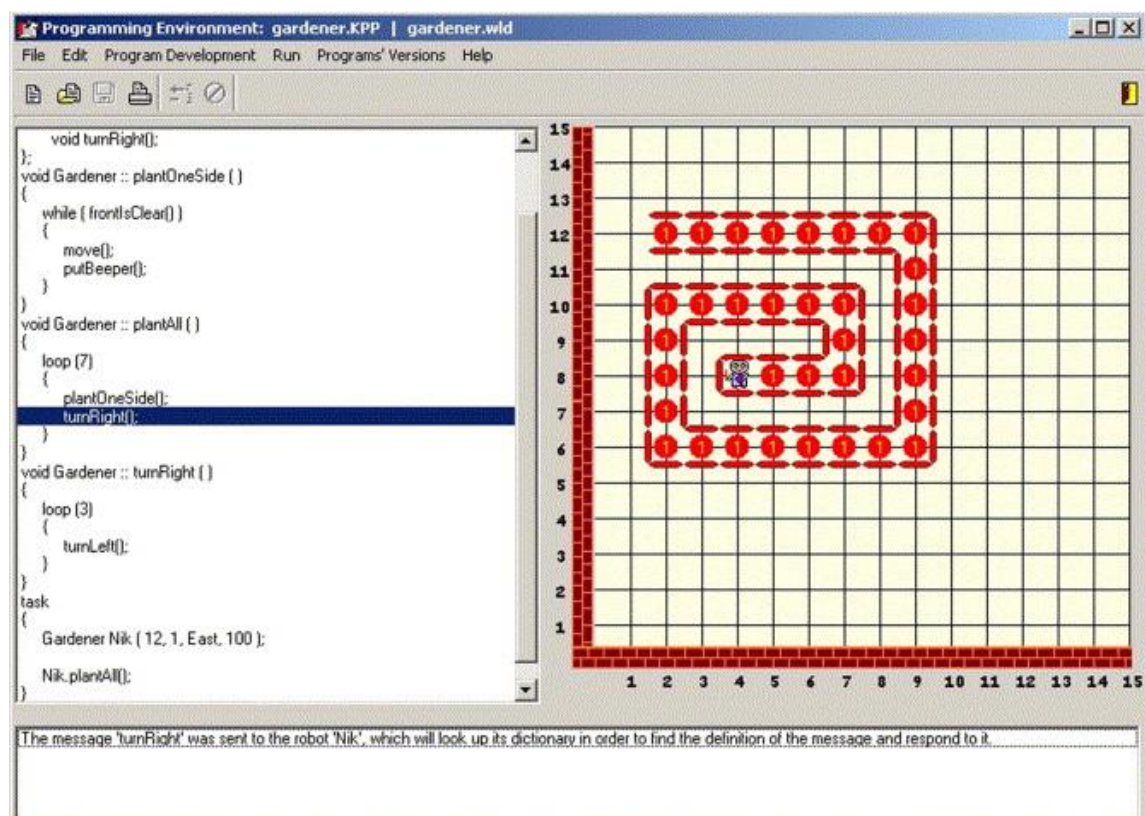


Figure 3.2: Programming environment for Object Karel (Xinogalos et al., 2006)

b. Raptor: An optic programming circumabient for tutoring object-oriented programming

Raptor is programming environment created to assist students in visualizing classes and methods by combining UML and flowcharts. Programs can be carried out manually and

then written as an object oriented programming language such as Java. The environment is designed in such a way that help students create algorithms by combining graphical symbols. This environment is good for novice programmers who possess little programming knowledge. In addition, the environment has built-in features for problem solving which are executed when algorithms are created visually as shown on Figure 3.3 below. After creating designs in Raptor, the output can be converted to Java language and the UML inventor allows students to create interfaces, teaching rooms and enumeration types. Comments can also be added on the UML diagram. The environment supports more than 80 built-in functions for students to generate random numbers, draw graphical designs, perform trigonometric computations and generate random numbers (Carlisle, 2009).

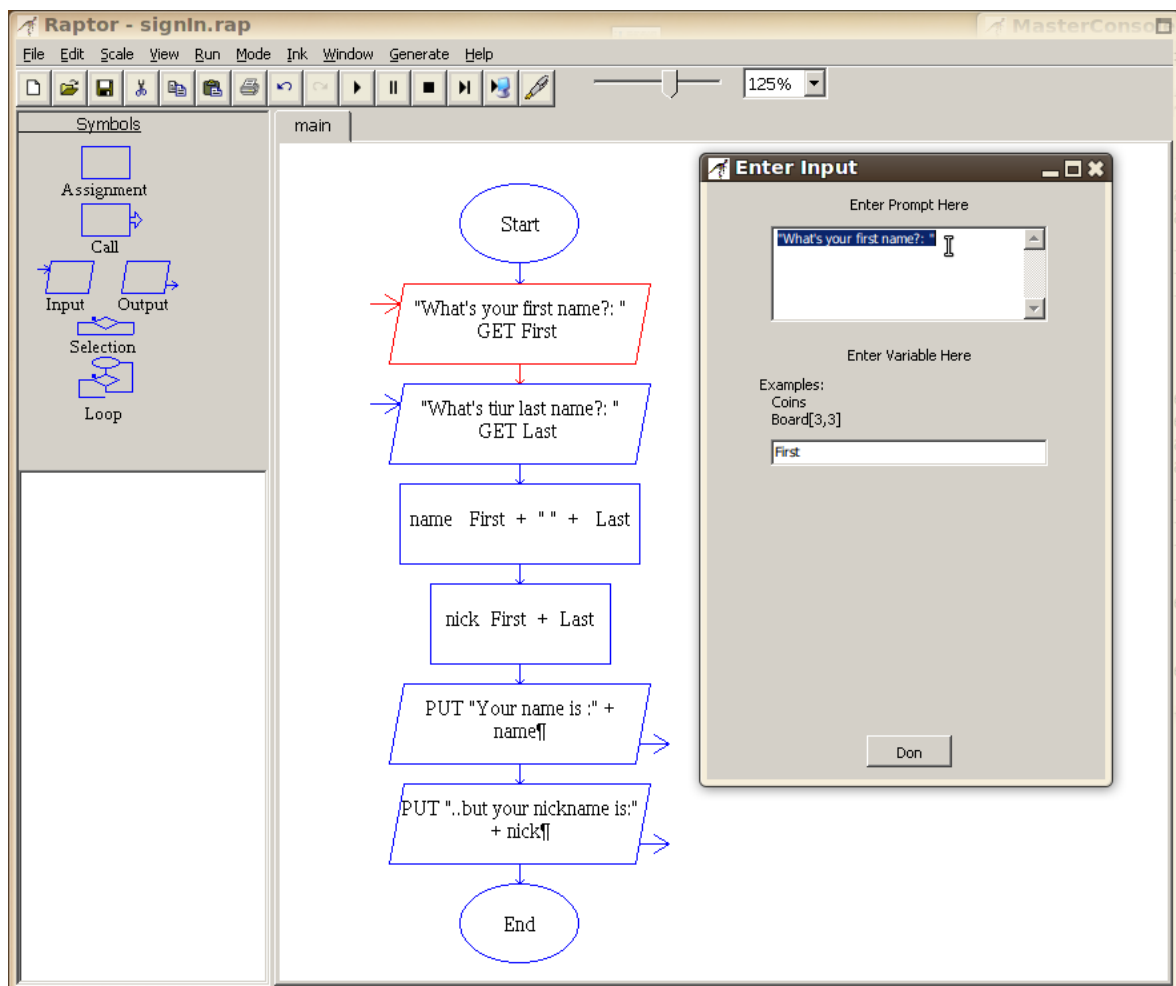


Figure 3.3: Screenshoot of Raptor visul programming environment (Harrykar, 2015)

3.7 Programming Learning Strategies

In the literature, many researchers (Chetty & Westhuizen, 2014; Kunkle, 2010; Young & Fry, 2012) have described the three main approaches that can be used as a teaching aid for introductory object-oriented programming. One of the approaches focus on writing source code whilst the other two approaches aims at establishing and on creating and shapping various classes and objects. The approaches explained by Chetty and Westhuizen (2014) are explained as follows:

a) Objects-First Approach Using BlueJ

BlueJ is an integrated development environment that allows novice programmers of Java to interact and visualise with classes and objects before writing lines of code (Barnes et al., 2017; Valdecantos, 2017). This approach introduces students to concepts such as classes, objects and methods at the introductory lessons and encourages students to interact with methods and objects before writing code. BlueJ was developed to run on Sun Microsystems' Java 2 Standard Edition (J2SE) Development Kit and it is an integrated development tool for java programmers. Kunkle (2010) pointed out that, by using BlueJ the shortcomings of many development environments are addressed which are listed below:

- The object-oriented paradigm is not reflected in other development environments.
- Most development environments are too complex for novice programmers and this makes learning programming difficult for novice programmers since the environments are designed for professionals.
- Other development environments focus mainly on building Graphical User Interface at the expense of allowing users to interact with objects and classes.

To address the above mentioned shortfalls of existing object-oriented development environments, the BlueJ development environment was developed (Kunkle, 2010). The following are the features of the BlueJ environment:

- BlueJ enables students doing introductory courses in object-oriented programming to interact with objects and classes making it easier for them to visualise what happens when programs are executed.

- Embedded UML diagrams support visualization and pop-up menus as well as dialogue boxes enable interaction between the environment and the user templates are also used to support easy code generation for novice programmers.
- BlueJ allows instructors to shield/hide more advanced concepts from students until a suitable time when the instructor feels it's now the right time to introduce these concepts.

A screenshot of BlueJ is shown on Figure 3.4 below. The screenshot is one of the default projects named “people” that is embedded in the development environment.

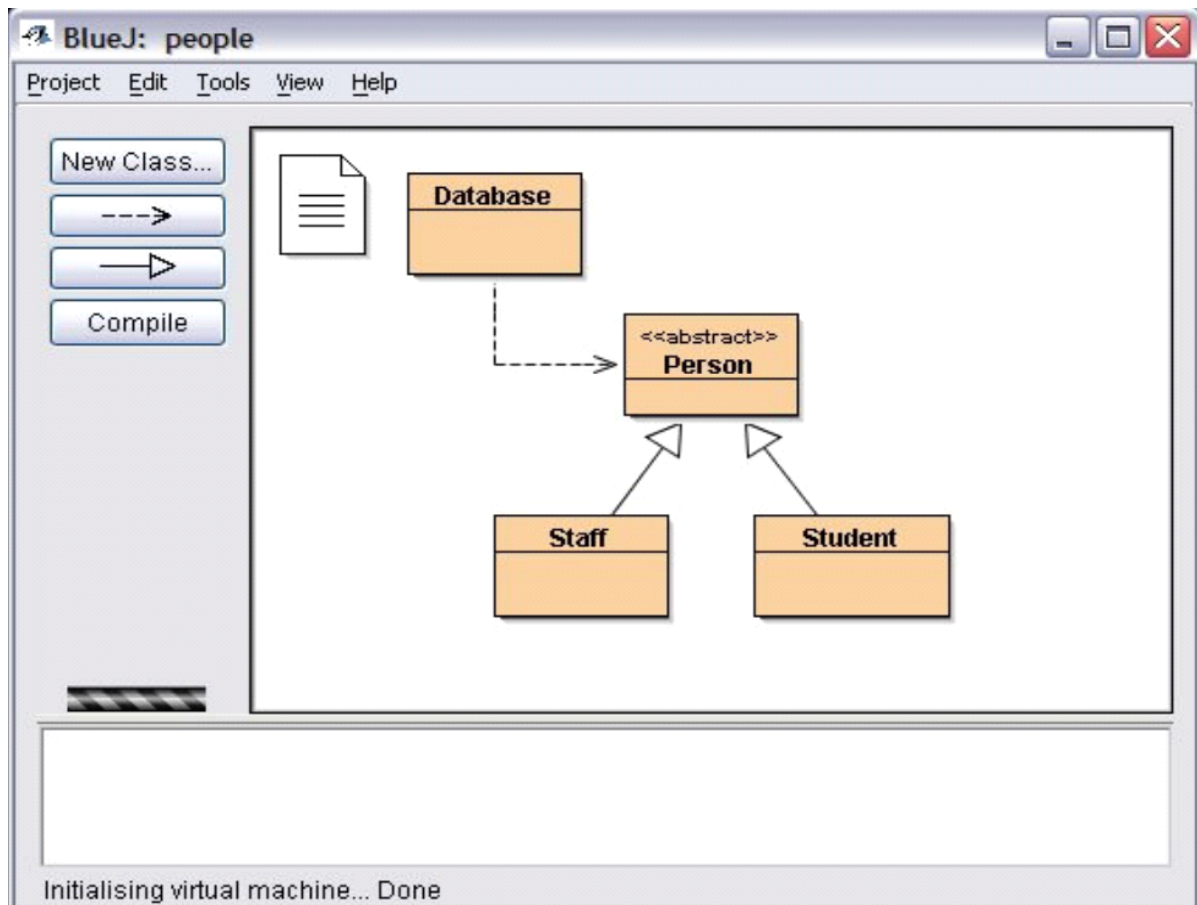


Figure 3.4: An example of BlueJ project people (Kunkle, 2010)

BlueJ is a tool that enables students to visualize what happens during program execution by developing a software environment that enables students to interact with objects and classes by building Graphical User Interface.

b) Objects-First or Objects-Early Approach Using Alice

This second approach uses a development environment called Alice. Alice allows students to build virtual world using 3D graphics by dragging and leaving objects, techniques as well as restrained structures (Alice, 2015). The distinction between the “objects-first” approach and the “objects-early” approach lies in the timeframe that is taken before students are actually introduced to coding. Objects-first has a longer time frame compared to the objects-early approach. Alice was initially designed for undergraduate students with no programming and 3D experience. Alice has enhanced students’ interest in object-oriented programming by providing the following:

- ***Less frustration:*** Syntax errors are minimised by allowing students to create programs in 3D without typing a single line of code. Programs, often referred to as worlds in Alice are created by dragging and dropping objects and methods.
- ***Friendly environment:*** Novice programmers are provided an environment where they can create compelling programs. Computer programs that take the form of 3D movies and games can be developed by using the storyboard as a metaphor and students can test the programs at any point during the design phase.

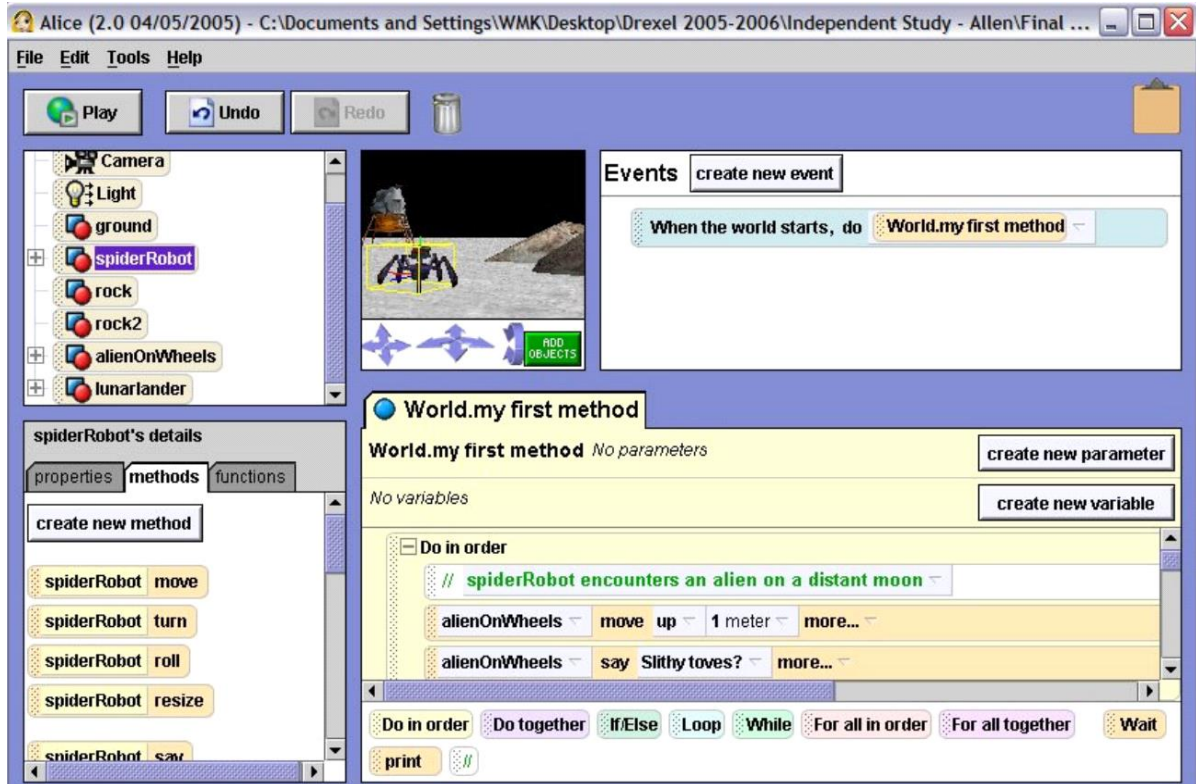


Figure 3.5: Shows a screenshot of Alice development environment (Alice, 2015)

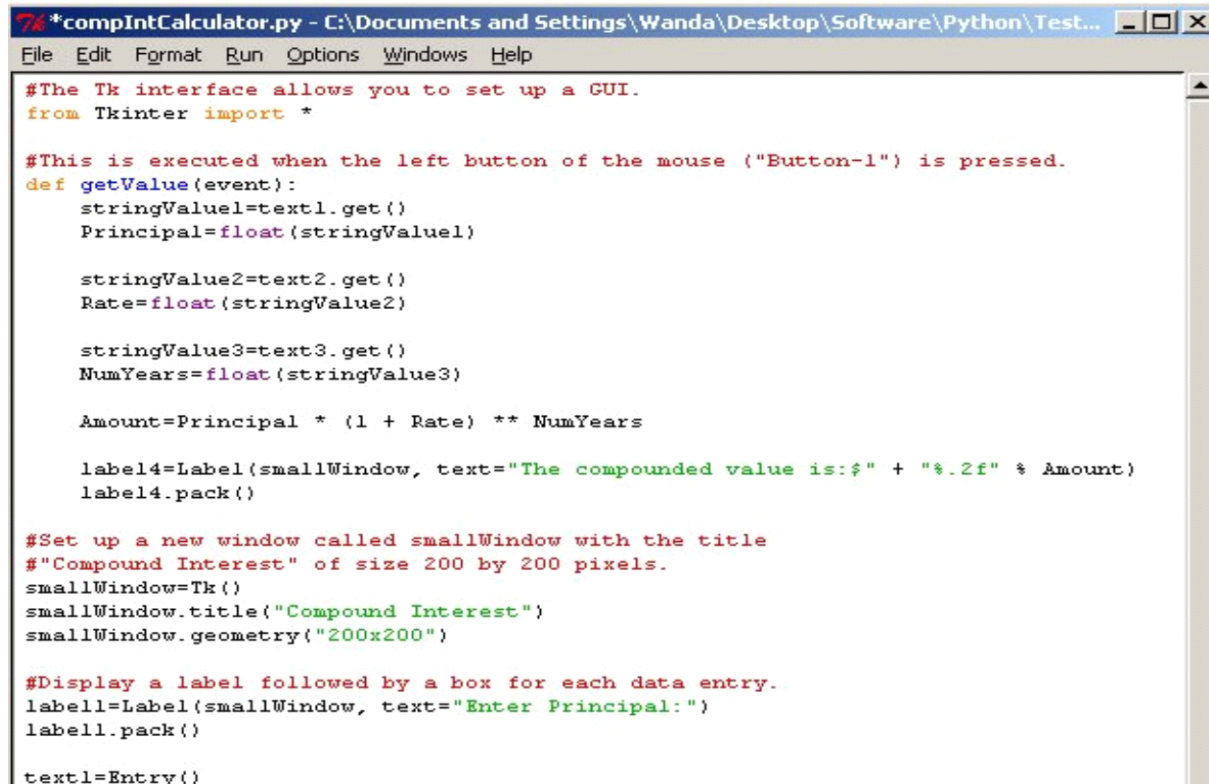
In studying about programs most students have difficulties learning to program, most students problems with creating algorithms, discovering out how to use problem solving techniques in their programs, and making use of simple programming tools. The Alice, a 3-D tool interactive animation environment. A new construct that prepares a feasible approach to actively involving students in increasing their knowledge in many areas

Alice (2015) is an inventive block-based programming environment that enables user to create with easy animations, design interactive narratives, or program simple games in 3D. Contradicting many concept of the puzzle-based coding applications, Alice inspires learning through creativity expedition. Alice was built to teach logical and computational thinking skills, basic principles of programming and exposing first object-oriented programming. The Alice Project demands use of supplemental skills and materials for tutoring using Alice across a spectrum of ages and subject matter with theoretical benefits in indulging and retaining different and underserved groups in computer science education (Costa & Miranda, 2017).

c) Imperative-First Approach Using Python

This approach has a traditional background in that it introduces programming first using python. Python is well known as an object-oriented scripting language used for introductory programming courses due to its nature of being simple, it provides flexible syntax and supports feedback (Kunkle, 2010). Students are introduced to imperative aspects first, which include expressions, control structures, functions and procedures that are the basics for procedural programming. Once mastered, students are then introduced to object-oriented techniques (Biju, 2013). Python is used when embedded in a host environment such as a web page. Below are the features of python that make the language more appealing for beginning programmers as stated by Kunkle (2010):

- Python uses simple syntax that is appropriate for beginners.
- Programs in python are enforced allowing the python container object to hold objects of any type or list.
- Python provides dynamic typing of variables.
- Python has embedded built-in types such as lists.
- Provides a friendly Graphical User Interface
- Provides an interactive mode where the user can experiment with the code.

A screenshot of a Python IDE window titled "compIntCalculator.py - C:\Documents and Settings\Wanda\Desktop\Software\Python\Test...". The window contains Python code for a Tkinter GUI. The code includes comments explaining the Tk interface, a function to get input values and calculate the compounded amount, and the setup of a small window titled "Compound Interest" with labels and entry fields. The code is as follows:

```
#The Tk interface allows you to set up a GUI.
from Tkinter import *

#This is executed when the left button of the mouse ("Button-1") is pressed.
def getValue(event):
    stringValue1=text1.get()
    Principal=float(stringValue1)

    stringValue2=text2.get()
    Rate=float(stringValue2)

    stringValue3=text3.get()
    NumYears=float(stringValue3)

    Amount=Principal * (1 + Rate) ** NumYears

    label4=Label(smallWindow, text="The compounded value is:$" + "%.2f" % Amount)
    label4.pack()

#Set up a new window called smallWindow with the title
#"Compound Interest" of size 200 by 200 pixels.
smallWindow=Tk()
smallWindow.title("Compound Interest")
smallWindow.geometry("200x200")

#Display a label followed by a box for each data entry.
label1=Label(smallWindow, text="Enter Principal:")
label1.pack()

text1=Entry()
```

Figure 3.6: Screenshot of a source code for calculating compound interest (Agarwal, 2005)

3.8 Problem Solving Approach

In the literature many researchers (Krunkle, 2010; Donchev & Todorova, 2008; Kirsti, 2004) defined an algorithm as a method for solving problems that consists of a defined set of instructions usually written in ordinary language. This process of coming with an algorithm is usually a trial-and-error process. If there are more ways of solving the same programming problem it, also means there are many algorithms. The choice of an algorithm to use is dependent on several factors such as accuracy, reliability and ease of modification (Krunkle, 2010). The algorithm that takes the least time to execute is considered the best algorithm to use.

3.8.1 Steps in problem solving

The section below explains the steps that are crucial for solving problems using algorithms. The steps below describe checks that must be done before an algorithm is executed to make sure that it exhibits certain properties. Below are some steps in problem solving as described by Kirsti (2004):

- **Generality:** An algorithm must be built for a full class of problems not just instance of a problem. It is crucial to check and verify input specifications if they meet the requirements and format that the computer will be able to execute. In addition, output specifications should also be checked to see if it complies with the format and make of the computer.
- **Finiteness:** All algorithms to be executed must have a finite structure that means that it must terminate with either the right output or an indication that there is no solution. Since computer algorithms repeat instructions, it is very important for finiteness to be considered.
- **Non-ambiguity:** All operations within an algorithm should be precise which means that the algorithm must be clearly stated and one has to know which step follows next. Commands used during execution may be conditional or repeated which mean they execute in loops, however for repeated commands, the algorithm should be designed to know when to stop.
- **Efficiency:** Memory and time resources should be taken into account for the algorithms to be considered useful. Useful algorithms are the ones that require a reasonable amount of computing resources.

3.8.2 Problem Solving Strategies for Programmers

Chetty and Westhuizen (2014) explained a sequence of steps that one out to follow in solving programming problems. The steps explained are described in detail below:

1. **Defining or specifying the problem:** It is crucial for one to understand the problem before attempting to solve it. The student should ask him/herself the following key questions; what will the computer program do, what tasks must the program perform, what will be the output of the program, what kind of data will be used and how will the program interact with the computer user.
2. **Analysing the problem:** The idea behind analysis is to find an appropriate solution to solve the problem. This phase involves identifying inputs, outputs and other requirements or constraints before attempting to solve the problem.
3. **Algorithm development:** During this phase an algorithm is designed, that rectifies the problem through the use of a pseudocode or a flowchart. A pseudocode describes the logic and flow of the program and is written in a plain language that

can be understood by the user. Alternatively, a flowchart can be used during this phase as it provides graphic symbols and arrows that aid in expressing the algorithm.

4. **Coding:** This refers to the process of converting an algorithm into a programming language that can be understood by a computer.
5. **Testing and Debugging:** Testing refers to the process of executing functions by entering data to check output whereas debugging is a process of finding and correcting program errors that are produced during program execution. The errors that are corrected during debugging are syntax flaw, run-time errors and logic errors.
6. **Documenting:** It is important to document the program for prospective reference. An internal and external document should be kept with a detailed explanation of how the problems were solved. This serves as a good reference point for future use and program modification.

3.9 Metacognitive Skills

Geiwitz (1994) defined metacognitive skills as one's ability in monitoring and directing different operations of coactive skills in order to gain the biggest possible success. In addition, the researcher defined a cognitive skill as the usage and manipulation of elements that enhance performance. Furthermore, he described problem solving as the generic cognitive skill. For problem solving to be a success, the researcher explained 10 metacognitive skills that lead to effective problem solving as follows:

- Discovery of a problem
- Illustration of a problem
- Adopting of a problem-solving method
- Strategic usage of problem solving methods
- Assessment of solution candidates
- Recognition of errors
- Resource distribution
- Physical monitoring
- Social monitoring
- Directing monitoring

Metacognitive is generally described as monitoring and controlling activities of one's cognition. Metacognition has been identified as an important factor to be a successful learner in learning computer programming Ramesh (2014). According to Flavell (1979) metacognition refers to knowledge about thinking and coactive phenomena and has the capability to produce deep and important learning (Garrison & Akyol, 2015; Siswati & Corebima, 2017). Metacognition includes the capability to think about mental activities, strategies and tasks, implement progress to direct and support coactive thinking, and showing on all actions successful, with the goal of establishing deep and important learning (Flavell, 1979; Rizk, 2017).

The reason of metacognition is to direct thinking in a manner that it effectively controls mental activities specially when handling real life problems and complex tasks. A distinction is observed among metacognitive knowledge and metacognitive control of experiences. Metacognitive knowledge includes knowledge of a person, knowledge of distinctive strategies and knowledge of task to complete a task successfully, while metacognitive control refers to managerial processes to reflect, plan, monitor, reflect on and evaluate activities such as critical thinking and problem solving (Titus & Annaraja, 2011):

- **Planning** is related with goal setting, reading and evaluation of text and of job to support understanding.
- **Monitoring** include an individual's attention of state of coactive activity, the skill to consult all detailed activities involved (e.g. problem solving), and the ability to assess progress.
- **Reflection** as part of metacognition demands that learners should reflect in action, that is, while doing a task and on action, that is, after completing a task.
- **Evaluation** is a metacognitive skill that explains the efficiency at which the task was performed and whether the main goals had been achieved.

3.9.1 Metacognitive skills for programmer

Young and Fry (2012) carried out an investigation to find the relationship that exists between metacognitive knowledge and metacognitive regulation, which was measured at a local and global level. Findings revealed that it is crucial to measure metacognitive

awareness since it helps in predicting a learner's academic success, learners can reflect on their learning skills and it indicates learners, which need further assistance.

The enhancement of programming solutions uses a combination of precision and artistic dexterity to create the desired program or software product. The need of metacognition in the answering of programming problems is imperative (Safari & Meskini, 2015). To do well in programming, the usage of more metacognitive management strategies are better than lower-performing strategies. The more complicated a programming problem is, the more the demand for metacognitive control, purposeful reflection and positive feedback (Havenga, 2011). Most programmers usually apply in-depth reading skills and meta-comprehension to determine how clear and efficient the programming problem is (Ramesh, 2014). In addition, programmers need to direct their problem-solving progress, decide on a programming approach to apply, correct programming errors, research about their programming solutions and test program output (Anandaraj & Ramesh, 2014). These steps for problem-solving (PS) demand metacognitive control such as monitoring the design, outlining the solution, and enhancement of the program and testing and reflecting on the programming solution (Safari & Meskini (2015). Furthermore one should develop their programming processes, predetermine their decisions, disperse their actions and inspect alternative solutions to enhance the quality of their programs. The programmer is responsible in developing metacognitive skills and applying these during problem solving and program development.

Moreover beginners face difficulty due to lack of experience and lack of knowledge of the object-oriented (OO) approach (Ginat & Shmollo, 2013; Sajaniemi et al., 2008), it is essential in this regard by applying more metacognitive thinking. Most researchers believe that the more complexity of a programming problem is, the bigger the need for metacognitive. Inasmuch as programming is complicated, it needs extra thinking skills that exceed just knowledge and codification. Major problems focused on the product of programming mainly on computer programs, instead of process of programming. This is proof from the experiences in class, as beginner usually start to program without having an overview of the problem and planning their solution. The programming usually focuses on results in repetitive actions of debugging, changing of code and addressing of some error messages, if possible (Sajaniemi et al., 2007; Stevenson & Wood, 2017).

3.9.2 Metacognitive steps for problem solving

Geiwitz (1994) defined metacognitive skills (MS) as one's ability in monitoring and directing different operations of coactive skills in order to gain the higher success. In addition, the researcher defined a cognitive skill as the usage and manipulation of elements that enhance performance. Furthermore, he described problem solving as the generic cognitive skill. Also for problem solving (PS) to be a success, many researchers (Safari & Meskini, 2015; Erol & Kurt 2017; Geiwitz, 1994) have explained 10 metacognitive skills that lead to effective problem solving as follows:

- ***Detection of a problem:*** Safari & Meskini (2015) pointed out that the ability to recognize a problem requires a special skill and is correlated with intelligence and creativity hence it varies among different people. For one to be able to detect a problem, the individual must be in a position to monitor any disparity between the present state and the stated goal and identify the problem once stated values are exceeded or not reached. Furthermore, the researcher pointed out that intelligent problem solvers, not only identify problems but they are also able to rank the problems from the least critical to the severe ones.
- ***Representation of a problem:*** After experienced problem detectors have recognized a issue, they structure the problem as such, it that makes is solvable (Safari & Meskini, 2015) The problem can either be represented using flow charts or in any other form that makes it easier to understand and come up with a solution to remedy the problem. This skill is important as proper representation of the problem may affect its result at the end.
- ***Choosine of a problem-solving method:*** There are many ways of killing a cat, so they say when it comes to problem solving (PS). Good problem solvers have a variety of methods they use to effectively solve problems. Special skills are required in selecting which solution best addresses the problem at hand.
- ***Strategic application of problem solving methods:*** Effective problem solvers use different strategies in answering problems effectively. They constantly monitor the changes in the problem to see if they have arrived at the desirable solution.
- ***Interpretation of solution candidates:*** Effective problem solvers use evaluation skills in assessing if potential solutions have been fully achieved. Evaluation of all

possible solutions must be done before selecting the most suitable one that fully addresses the problem at hand.

- **Recognition of errors:** The ability to spot errors is a skill. Good problem solvers are known to identify errors quick and more spontaneously. It is crucial to guard against common areas that result from misapplied heuristics and cognitive biases.
- **Resource allocation:** The ability to fully allocate resources is crucial in order to create the best environment for problem solving. Good problem solvers allocate resources wisely, for instance, when the nature of the problem requires memory as a resource then good problem solvers are able to determine how much time will be required to memorize the material and by so doing they are able to determine the duration.
- **Short monitoring:** This refers to efficient time allocation and monitoring skills to see if the solution is aligned with the proposed schedule. Erol and Kurt(2017) pointed out that effective managers are known to use their time effectively and are good at managing their schedules. For maximum impact when dealing with complex problems, it is crucial to synchronize resources.
- **Social monitoring:** This refers to problem solving (PS) in a social context whereby human resources are allocated in such a way that a social environment is created whereby individuals can interact as they solve problems. Different personalities should be taken into considerations before grouping people to avoid unnecessary disputes and understand the different goals to be achieved. Interpersonal relations are the most important when it comes to solving problems among individuals in groups.
- **Executive monitoring:** For effective problem solving it is critical for executives to fully understand their position on the organizational structure. This includes their roles, relationship with subordinates, peers and higher authorities. It is crucial for executives to be able to have marketing skills and good knowledge of how the company operates. Such marketing skills develop over time and most executives never become fully accomplished at business development.

3.9.3 Metacognitive components for problem solving

In the literature, many researchers (Safari & Meskini, 2015; Anandaraj & Ramesh, 2014; Sternberg, 1984) have described the various components of metacognition that are important when dealing with problem solving. The researchers point out the importance of organizing, sequencing and monitoring cognitive processes to achieve success. Also known as meta-components, these are high level effort used in creating, observing and calculating one's ability to solve problems. The researchers came up with the following seven components listed below and Geiwitz (1994) also explained in Figure 3.7 below:

- Decide the problem
- Select the lower-order components of solving the problem
- Select information representations
- Select a strategy for combining lower-order components
- Decide the allocation of resources
- Monitor the result
- Observe Result

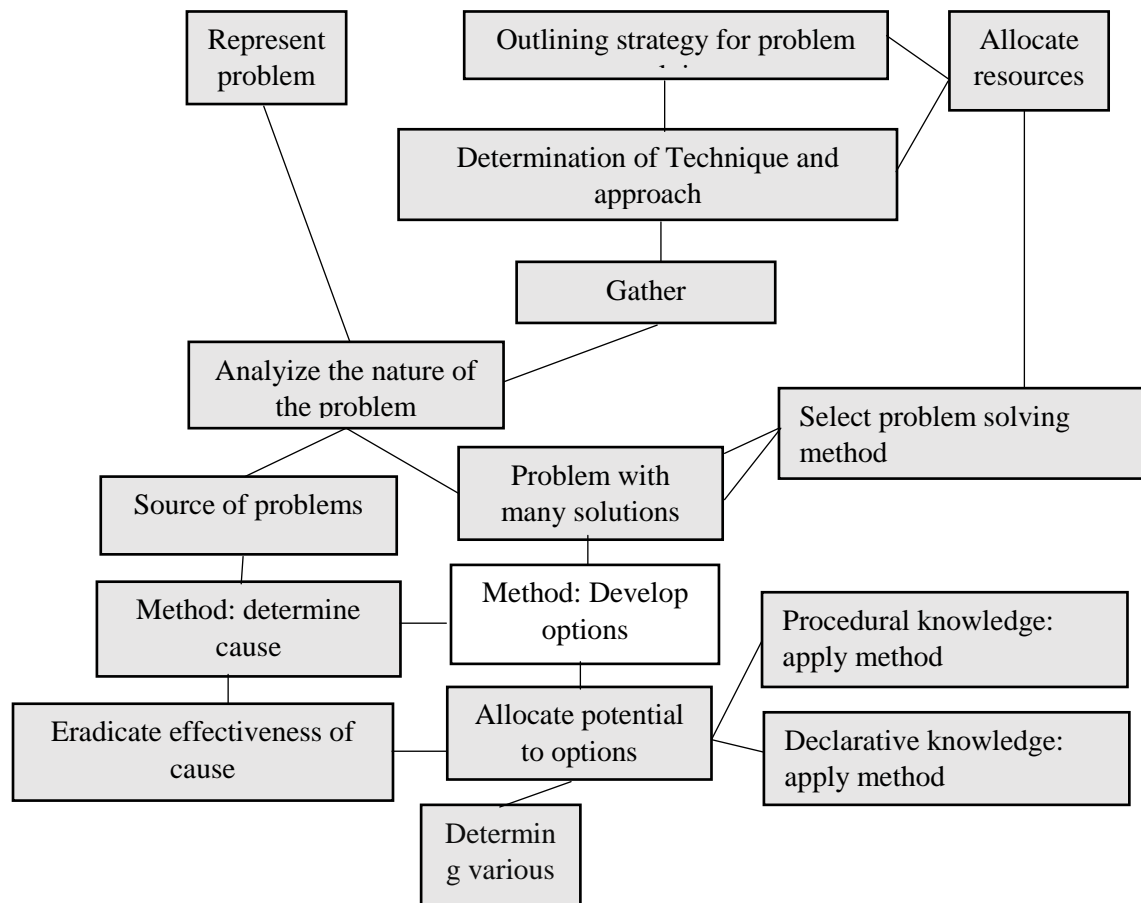


Figure 3.7: Stages in metacognitive skills for solving programming problems (Geiwitz, 1994)

In Figure 3.7 Among the metacognitive skills involved in planning, effective and efficient allocation of resources is one that seems to be required at several points in the problem solving process such as Caused problems are the type for which the methods are appropriate, that is, something is causing a problem for a business, and the identification of the cause is accomplished by comparing similar situations with and without problems. Problems with multiple solutions include the problems faced by mission planners, who use the general method of generating several options. Plan strategy for problem are. Allocate resources.

Safari and Meskini (2015) explained four metacognitive skills that are essential in problem solving. Firstly, the problem must be identified and defined. Secondly, the problem must be represented in such a way that it can be understood clearly. Thirdly, a plan should be put in place on how the problem will be tackled and lastly, performance evaluation must be

done to understand how the solution will be reached. Below in Table 3.1 are the metacognitive components as described by the researchers:

Table 3.1: Showing different metacognitive components and implications in problem solving (Safari & Meskini, 2015)

Metacognitive components	Implications for problem solving
Planning phase: Think deeply of all possible strategies (planning)	<ol style="list-style-type: none"> 1. Ask yourself if you have understood what is needed in solving the problem? 2. Disintegration (break down the problem into different sections for manageability) 3. Separate the different sections 4. Solve the problem components separately
Accomplish a strategy (control and monitoring)	<ol style="list-style-type: none"> 1. Ask yourself if you have determined the patterns and paths for solving the problem? 2. Ask yourself if you are able to write the possible solutions to get the correct solution? 3. Do you have enough time to solve the problem?
Revise and regulate	<ol style="list-style-type: none"> 1. Ask yourself if you are in the right path? 2. Are you approaching the goal? 3. Are you still using the strategies you selected? 4. Do you need to review the problem again and use another strategy? 5. Can you solve the problem using a different method? 6. Is the solution correct?

A model was developed by Geiwitz (1994) that illustrate the relationship that exists between different metacognitive skills involved in problem solving as illustrated in Figure 3.8 below. The model comprises of different essential steps from:

- problem detection
- representing the problem
- strategizing the best solution
- applying good methods
- implementing solutions
- monitoring the solution
- evaluating the solution to see if the anticipated result has been achieve

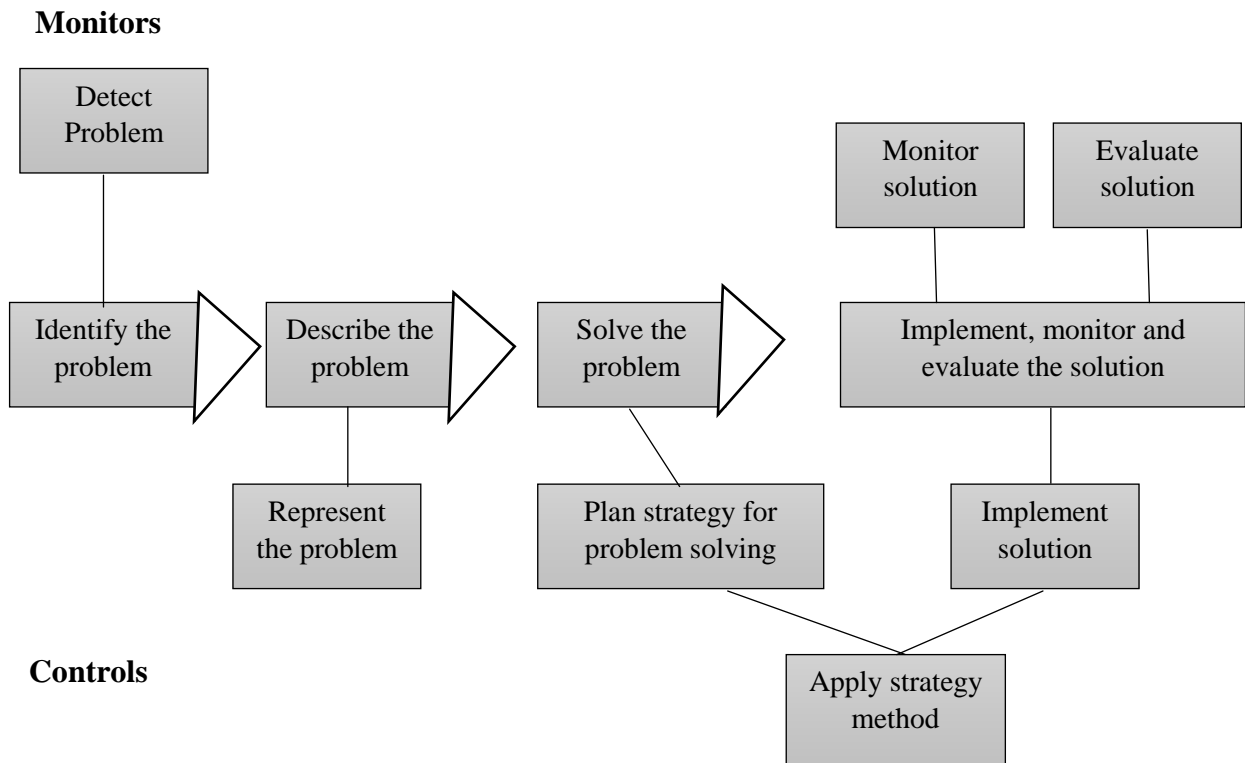


Figure 3.8: A model of metacognitive skills and how to effectively solve problems (Geiwitz, 1994)

Figure 3.8 tries to align the metacognitive skills relevant to monitoring and controlling the technical process of problem solving (PS). In its present evolution, the conceptual model of technical skills identifies seven major capabilities as shown above that focuses on skills that facilitate the technical activities in problem solving (PS). The technical aspects of problem solving comprise the purely formal operations designed to identify, represent, and solve the problem. In addition to the technical aspects, there are temporal, social, and organizational aspects of the problem solving (PS) process.

CHAPTER 4

RESEARCH METHODOLOGY

This chapter explains the research model, participants, sample selection, data collection and tools, data analysis and the research procedure. Reliability tests are also shown in this chapter for the survey tool used.

4.1 Research Model

The main aim of this study is to investigate the effect of metacognitive skills towards problem solving (PS) in programming learning focusing on four Cyprus universities. Figure 4.1 illustrates the research model used in this study. The survey consists of independent variables: Metacognitive skills (MS) for computer programming. The dependent variables were problem solving skills. The research questions of the study have been taken from the figurative view of the research model illustrated below:

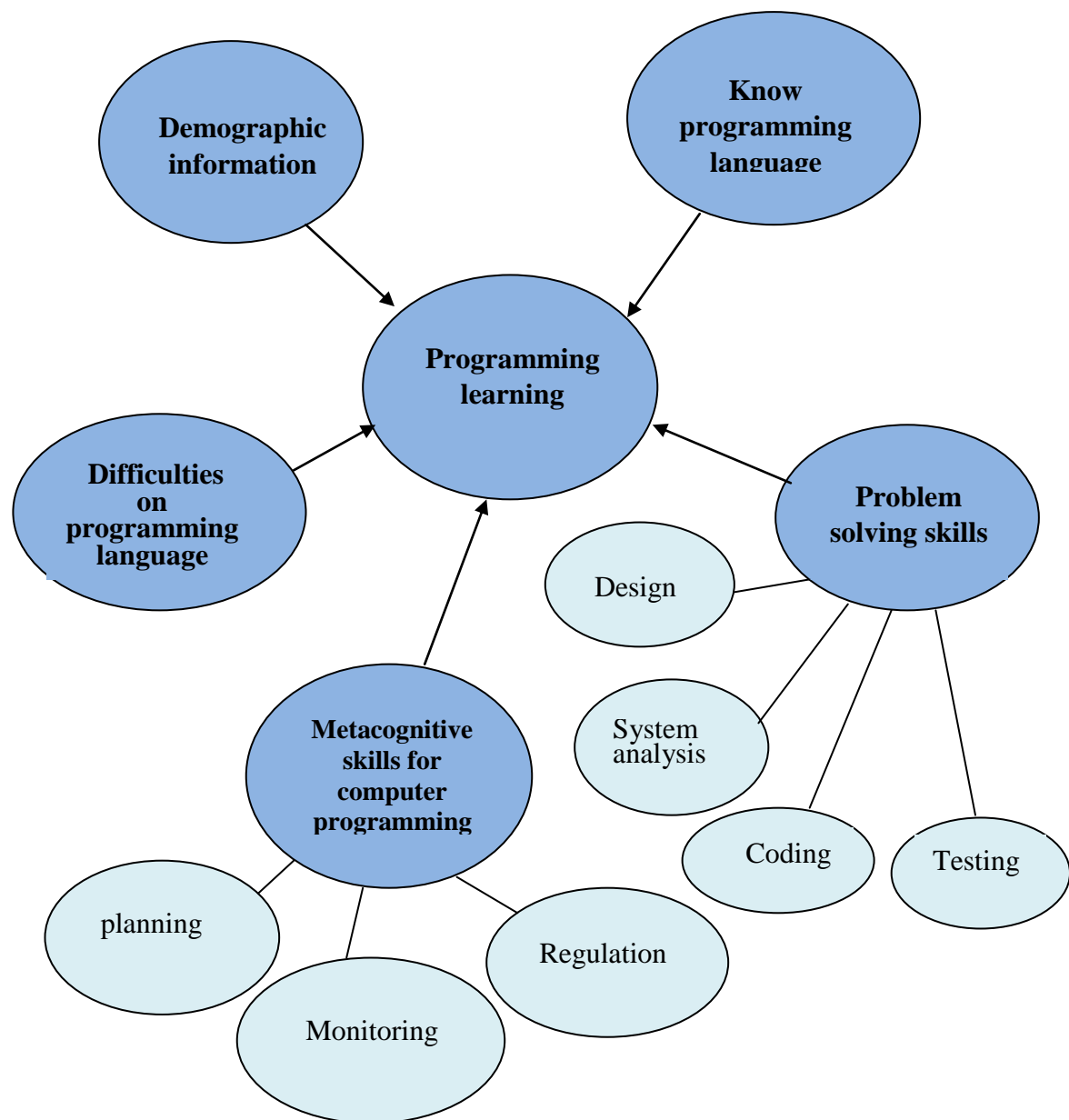


Figure 4.1: Research model of the study

In the survey method, the questionnaire was the best way to acquire data from a large group of respondents. The Researcher distributed 300 questionnaires on four universities affiliated with north Cyprus. In addition, the random sampling method was used for data collection on undergraduate, masters and PhD students of Computer Information System (CIS), Information Technology (IT), Computer Engineering (CE), and Management Information System (MIS).

4.2 Participants

The research targeted students currently enrolled in undergraduate, masters and postgraduate programs from the four universities in North part of Cyprus namely Near East University (NEU), Eastern Mediterranean University (EMU) Cyprus International University (CIU) and European University of Lefke (EUL). A total of 300 participants were chosen using the purposive sampling method. Ritchie et al. (2013) in this method, the researcher picked out a set of people among population under consideration based on research, which was made up of 110 undergraduate and 190 postgraduate students (Masters and PhD). These were taken from different departments namely Computer Information System, Information Technology, Management Information System, Computer Engineering whose students taken a program course. Using Raosoft sample size calculator, the population size from four departments was 1000 and the confidence level was 95% because the survey questions was more than 20. The Raosoft sample size calculator gave the minimum recommended size of the survey to be 278. For this Survey the targeted students was 300 which is approximable to the Raosoft calculator recommended size.

Table 4.1: Important about chosen universities

University name	Frequency	%
NEU	100	34
EMU	70	23
CIU	70	23
EUL	60	20
Total	300	100

Higher percentages were from NEU, EMU and CIU with values of 34%, 23% and 23% respectively. EUL was the lowest with 20% of undergraduate students. Among the Students who joined the study from both faculties, 56.84% were males and 43.16% were females. The characteristics of the respondents are presented in Table 4.2. Also as illustrated in the table below, 27% of the students were aged from 18-20, 33% from 21-24, and 40% were 25years and above (Table 4.2).

Table 4.2: Important demographic data of participants

Characteristic	Frequency	%
<i>Gender</i>		
Male	175	58%
Female	125	42%
<i>Age</i>		
18-20	80	27%
21-24	100	33%
25+	120	40%
<i>Degree</i>		
Undergraduate	110	37%
Postgraduate (Masters and PhD)	190	63%
<i>Faculty</i>		
Faculty of Engineering	167	56%
Faculty of Economics and Administrative Sciences	133	44%

4.3 Data Collection Tools

The questionnaire was in English language and composed of three parts as illustrated in Figure 4.2.

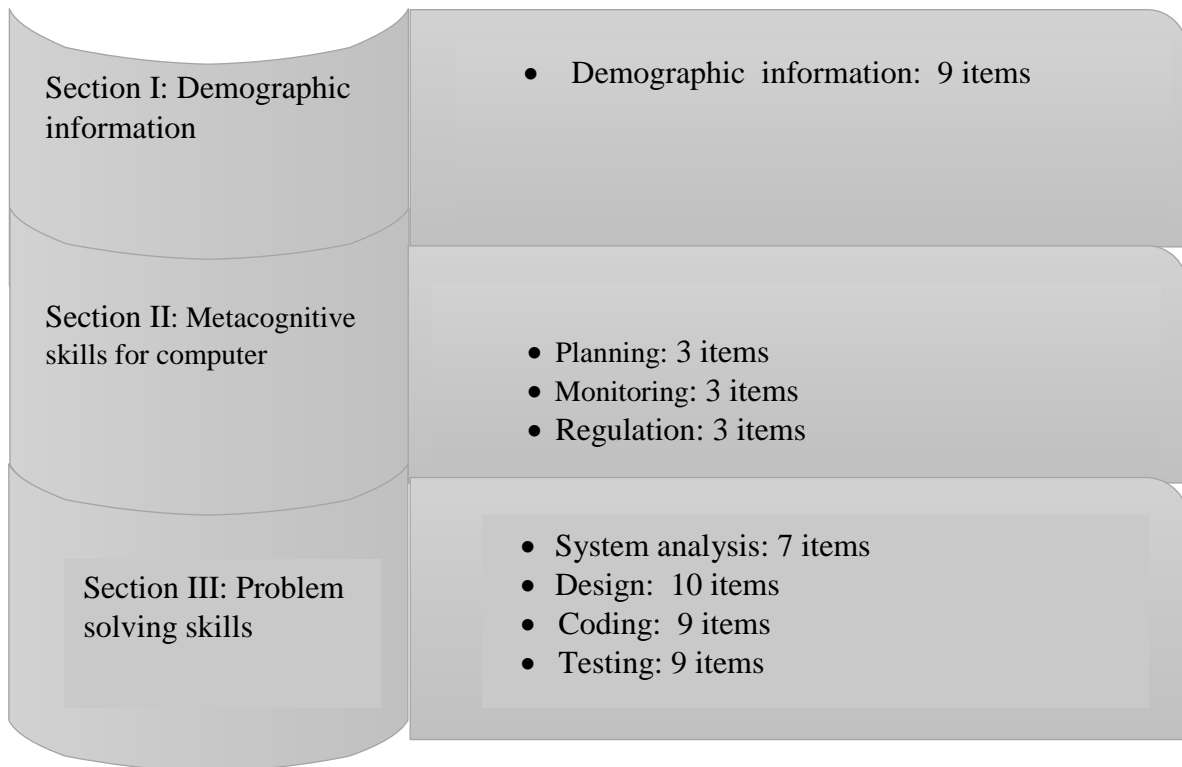


Figure 4.2: Structure of the questionnaire

Section I: Demographic Information: Requested the participant information such as gender, age group, level of study, university name, native language, department, operating system and programming languages that the students know and their level of programming skills.

Section II: Metacognitive skills for computer programming: The aim of this section was to gain an insight of the participants such as knowing the effect of metacognitive skills for computer programming. The questionnaire was adapted from Ellis and Stcyn (2003). This section comprised of three dimensions namely “*Planning*”, “*Monitoring*” and “*Regulation*” and it had 9 items in total. In “*planning*”, 3 items were assigned to it, whose goal was to search for the analysis of a programming problem, to identify the objects, methods and the application of a strategy necessary to solving the problem and

achieving the goal. In “*Monitoring*”, 3 items were assigned to it, whose goal was discover how the students used observing strategies in order to control their own programming activities. In “*Regulation*”, 3 items were assigned to addresses it. This was to determine whether students view regulation of the programming process as crucial. In this study, the Cronbach’s Alpha for the whole scale was calculated 0.80. Therefore proving that our scale is indeed reliable (Sipahi, Turtkoru & Cinko, 2010). With scales used in which participants were asked to assign a scale measurement based on the 4 Likert scale ranked with Always (1 point), Often (2 points), Seldom (3 points), Never (4 points).

Section III: Problem solving skills: The aim of this section was to concentrate thoroughly on problem solving techniques used in programming and recommend an integrated methodology that makes the combination of problem solving and programming. The questionnaire was developed by my thesis supervisor based on literature and used in the thesis. The section comprised of four dimensions namely; system analysis, design, coding, and testing. This had 32 items in total. In “*System Analysis*”, 7 items $\alpha=0.79$ were assigned to it, whose goal was to build a full understanding of the problem. In “*Design*”, 10 items $\alpha=0.84$ were assigned to it and the aim was to deduce a programming problem, using former designs (diagrams and cases), make analysis, comparisons of the solutions, categorize, combine, modify a design, interpret and evaluate the proposed design. In “*Coding*”, 9 items $\alpha=0.85$ were assigned to it whose aim was to have knowledge and interpret the design, using former solutions in a new program, analyze and make comparisons of the outcomes, categorize, combine, design and make the program better, interpret and evaluate the outcome. In “*Testing*”, 6 items $\alpha=0.85$ were assigned to it, whose goal was to understand and interpret a solution, analyze and compare solutions, construe and assess the thinking and problem solving. According to the findings, the reliability coefficient and the whole scale of the sub dimensions are above 0.70 and this clearly shows that the scale is very reliable (Sipahi, Yurtkoru & Cinko, 2010). The Cronbach’s Alpha for these dimensions in the scale were calculated 0.94 with ordinal scales used in which participants were asked to assign a scale measurement based on the 5 Likert scale ranked with “*Strongly Disagree*” (1point), “*Disagree*” (2 point), “*Neutral*” (3 point), “*Agree*” (4 points), “*Strongly Agree*” (5 points).

Table 4.3: Problem solving skills

Dimension	Items	Cronbach's alpha
System Analysis	7	0.79
Design	10	0.84
Coding	9	0.85
Testing	6	0.85
Total	32	0.94

4.4 Data Analysis

The questionnaires were used to collect the data. The data was then analyzed and interpreted using SPSS version 20.0. Descriptive analysis and Pearson correlation analysis were used during the analysis process.

The following table can be used for a better understanding of the knowledge level of the programming language that the students possess.

Table 4.4: Score boundaries of the 5 Likert scale of level of knowledge of programming Language

Value	Limit	Likert scale
1	1.00-1.79	Very good
2	1.80-2.59	Above Average
3	2.60-3.39	Average
4	3.40-4.19	Little
5	4.20-5.00	Don't know

The following table illustrates the knowledge level of students about topics in a programming language.

Table 4.5: Score boundaries of the 5 Likert scale of level of difficulties faced by student

Value	Limits	Likert scale
1	1.00-1.79	Very easy
2	1.80-2.59	Easy
3	2.60-3.39	Moderate
4	3.40-4.19	Difficult
5	4.20-5.00	Very Difficult

Below is a table that shows the knowledge level of students in problem-solving skills.

Table 4.6: Score boundaries of the 5 Likert scale of level of problem solving skills

Value	Limits	Likert scale
1	1.00-1.79	Strongly agree
2	1.80-2.59	Agree
3	2.60-3.39	Neutral
4	3.40-4.19	Disagree
5	4.20-5.00	Strongly Disagree

4.5 Procedure

Research procedure cycle includes the related literature and studies done by the researchers. This will also present the theoretical and conceptual framework and the lastly related questionnaires' of the study. Item pool: The items of questionnaires were formed by looking at a set of survey lists dealt with in the previous studies and extracted a large number of questionnaire items that were related to the study. Reliability test determined how closely related a set of items are as a group using that Cronbach's alpha. A pilot study was conducted to check the feasibility of the study and to do necessary modifications of questions, validity, and reliability of the questionnaire. IT expert view determined the language that was used and the content. For data analysis and interpretation, Descriptive Statistics and Pearson Correlation were used and also for report writing.

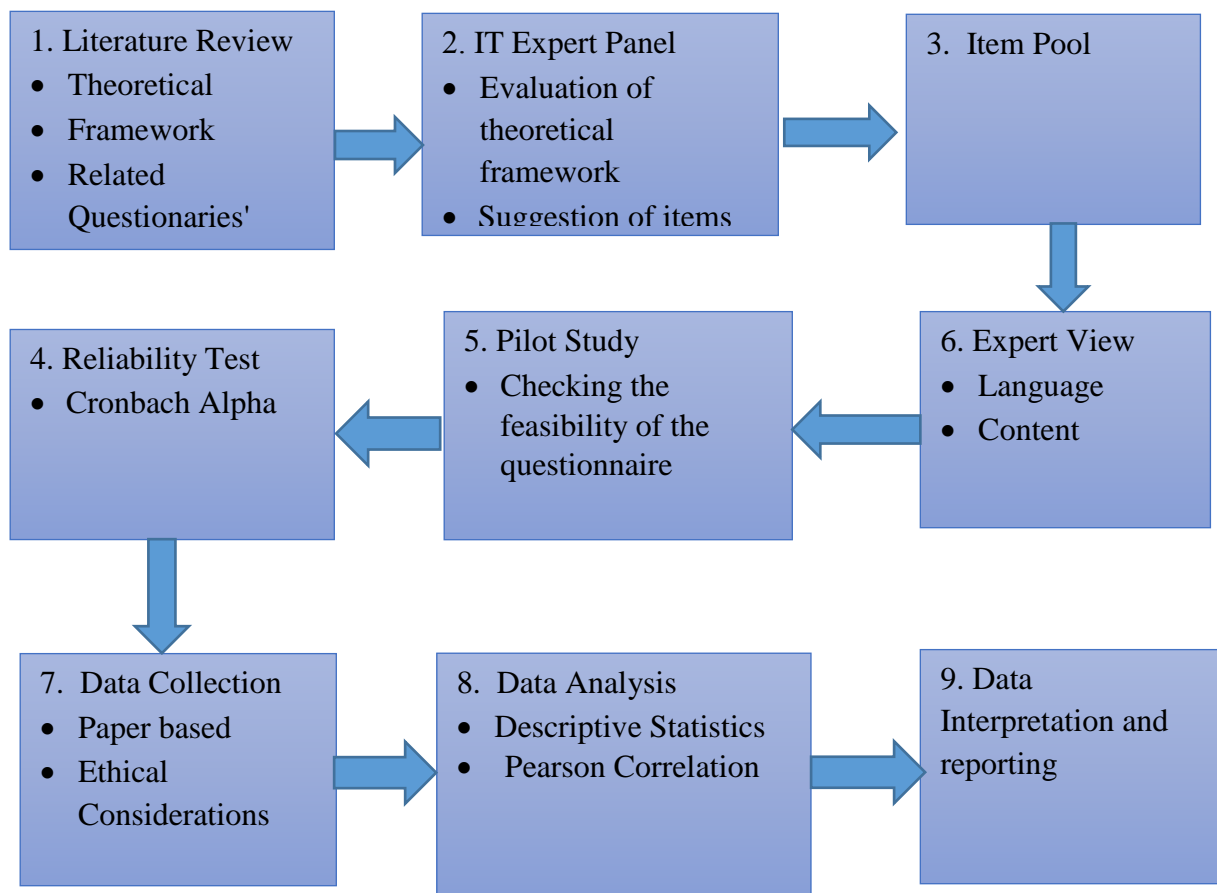


Figure 4.3: Research procedure

4.5.1 Ethical Considerations

In order to carry out the research from four universities in North Cyprus, the Ethical Committee for Scientific Research requested that some ethical considerations should not be exceeded in this research. These include the confidentiality of information for participants as well as the non-representation of the preliminary results of the data in a biased manner. In addition, the data was to be collected specifically from university students in computer science departments who study programming languages. Since the survey was for computer science students, students who opted to participate are the ones whose data will be analysed. Prior to the survey, the researcher informed all participants the nature of the survey and that it was voluntary and responses were confidential and were to be used for educational purposes only. A sample of the ethical letter is shown in appendix 2.

4.6 Research Schedule

This study started in November 2016 and was completed in October 2017. Figure 4.5 gives a detailed description of the tasks and its duration during the course of the research.

Figure 4.6 shows the Gantt chart for the thesis. All expenses incurred during the study were financed by the researcher, hence they have been excluded in the schedule on Figure 4.5.

Task Name	Duration	Start	Finish	Predecessors	Assigned To	% Complete	Status
1. thesis proposal and review	38d	11/01/16	12/22/16		Rajaa Jawadi		
1.1 identifying research area	29d	11/15/16	12/23/16		Rajaa Jawadi	100%	
1.2 searching the literature	7d	11/22/16	11/30/16		Rajaa Jawadi	100%	
1.3 formulating research question	4d	12/22/16	12/27/16		Rajaa Jawadi	100%	
1.4 Writing research proposal	20d	11/22/16	12/19/16		Rajaa Jawadi	100%	
1.5 research proposal review	1d	02/01/17	02/01/17		Rajaa Jawadi	100%	
2. preparation and development	25d	01/01/17	02/02/17		Rajaa Jawadi		
2.1 drafting of the questionnaire	8d	01/02/17	01/11/17		Rajaa Jawadi	100%	
2.2 distributing questionnaire to	10d	01/05/17	01/18/17		Rajaa Jawadi	100%	
2.3 obtaining feedback from test	6d	01/22/17	01/27/17		Rajaa Jawadi	100%	
3. Survey data collection and	70d	12/22/16	03/29/17		Rajaa Jawadi		
3.1 Data collection	55d	12/22/16	03/08/17		Rajaa Jawadi	100%	
3.2 Data Analysis	15d	08/15/17	09/04/17		Rajaa Jawadi	100%	
4. Compiling Thesis Document	20d	03/30/17	04/26/17		Rajaa Jawadi		
4.1 Writing final draft of the Thesis	10d	03/30/17	04/12/17		Rajaa Jawadi	100%	
4.2 Document review and correction	10d	04/13/17	04/26/17		Rajaa Jawadi	100%	
4.3 final thesis review					Dr Nadira	100%	
TOTAL DURATION OF PROJECT	153d				Rajaa Jawadi	100%	

Figure 4.4: Research schedule of the study

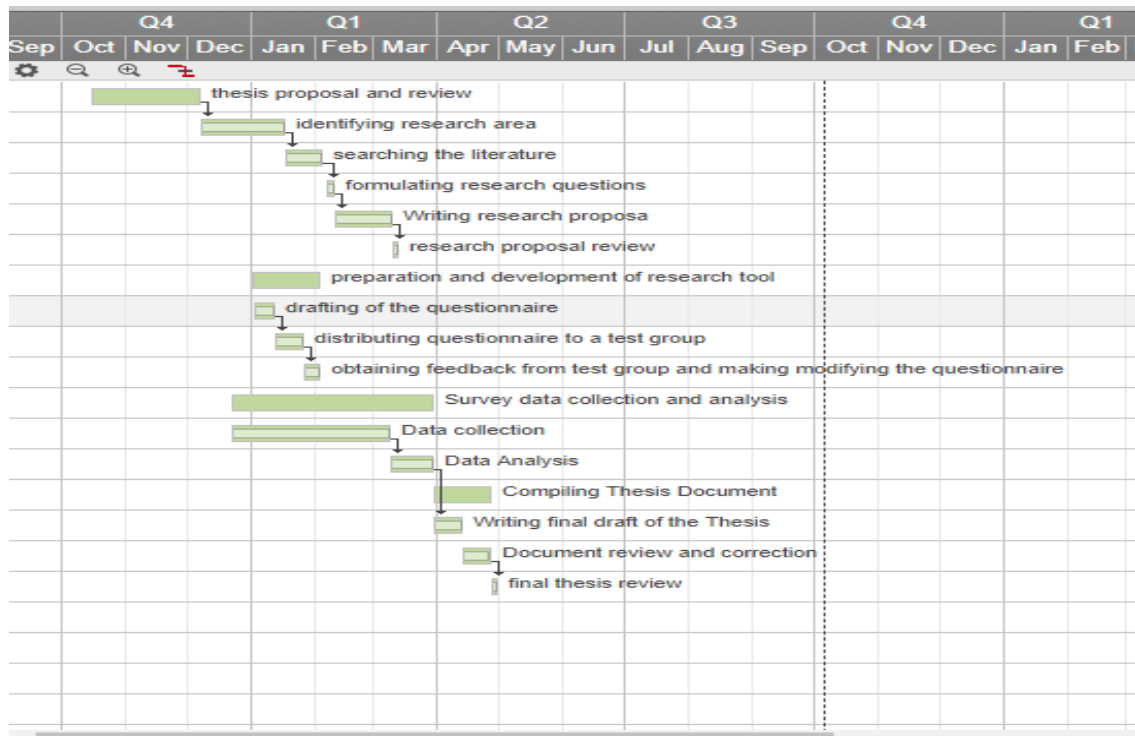


Figure 4.5: Gantt chart for the study

CHAPTER 5

RESULTS AND DISCUSSION

This chapter reflects on the results of the research in directed towards the original aims and questions of this research.

5.1 Programming Language Knowledge Level of Students

In order to understand the opinions of the students about knowing a programming language, descriptive analysis was employed. For script-based programming language, the knowledge level of students seems to be “*little level*” know and the average score is ($M = 3.81$; $SD = 1.07$). The Knowledge level of students on Object orientated programming language seems to be “*average level*” know and the average score for object orientated programming language is ($M = 2.77$; $SD = 1.10$). For the script-based programming language Ruby, it seems students do not have the knowledge about this programming language and for Pear it was “*little level*” know with; ($M=4.34$, $SD= 0.95$, $M= 4.00$, $SD=1.07$) respectively. For object orientated programming language .NET it was “*little level*” know with ($M=3.50$, $SD= 1.19$). Also for web-based programming language ASP it was “*little level*” know and PHP was “*average level*” know with ($M=3.53$, $SD= 1.18$, $M=3.04$, $SD= 1.13$) respectively.

Table 5.1: Mean and standard deviation programming language that the students Know

OBJECT ORIENTATED	Mean	SD
C++	2.18	1.01
C#	2.51	1.06
Visual basic	2.85	1.14
Java	2.82	1.08
.NET	3.50	1.19
Total average score of	2.77	1.10
WEB BASED		
ASP	3.53	1.18
HTML	2.63	1.11
Java Script	2.90	1.09
PHP	3.04	1.13
Total average score of	3.03	1.13
SCRIPT BASED		
C Language	2.96	1.22
Pascal	3.46	1.17
Pear	4.00	1.07
Fortran	3.97	1.00
Python	3.93	1.03
Delphi	3.98	1.03
Ruby	4.34	0.95
Total average score	3.81	1.07

5.2 Difficulties of Students on Learning Programming Language

Descriptive analysis was employed in order to understand the opinions of the students' level of learning and their difficulties in different topics of programming languages. We have to bear in mind that the responses are subjective, opinions of the students who answered the questions in the survey. The students might think that they understand the subjects of programming languages while in actual fact they might be difficult for them to understand.

Memory allocation operations seem to be on “*difficult level*” to be learnt by students, the results show that the total mean and standard deviation values for both items is ($M = 4.00$; $SD = 0.02$) which gave the highest mean value out of the total average score of all dimensions. The level for these dimensions are convergent and both are almost the overall average. These results can be attributed to the fact that as students’ progress in learning the subjects of programming languages, they face more challenges which require a greater understanding of the program. Similar results were also found in a study conducted by (Milne & Rowe, 2011). Most difficult topics are those related to Memory. Basic operations seem to be on “*Very easy level*” for students to learn. The results show that the mean and standard deviation values for all 5 items is ($M = 1.58$; $SD = 0.84$) which gave the least mean value out of the total average score of all dimensions since understanding them is not related to understanding the recognized “core” of programming. Basic operations seem to be very easy to learn from other dimensions, the results show that the average for all 5 items ranges from (1.25-1.91). Similar results were also found in a study conducted by (Milne & Rowe, 2011)

Table 5.2: Mean and standard deviation for items

Basic operations	Mean	SD
1. Variable/function declarations	1.25	0.64
2. Mathematical operators and precedence	1.40	0.74
3. Conditional operations (if, else, etc.)	1.57	0.83
4. Looping operations (for, while, etc.)	1.91	1.02
5. Input/output and file handling	1.79	0.96
Total Average score of 1-5	1.58	0.84
MORE COMPLEX OPERATIONS		
6. Arrays	1.76	0.92
7. Other data structures (trees, linked-lists)	2.21	1.06
8. String handling	2.64	1.20
9. Structures	2.69	1.21
10. Pointers	2.86	1.26
11. Recursion	2.91	1.16
Total Average score of 15-17	2.51	1.14
OBJECT ORIENTED OPERATIONS		
12. Classes and objects	2.90	1.16
13. Casting	3.07	1.18
4. Inheritance	3.15	1.07
15. Encapsulation	3.46	1.10
16. Polymorphism	3.57	1.08
17. Passing by reference/passing by value	3.66	1.10
18. Function overloading/default arguments	3.51	1.04
19. Function over-riding (in inheritance)	3.69	1.03
20. Constructors/destructors	3.78	1.01
21. Templates	3.84	1.03
22. Copy constructors	3.83	1.05
23. Operator overloading	3.85	1.05
24. Virtual functions	3.80	1.02
Total Average score	3.55	1.07
MEMORY ALLOCATION OPERATIONS		
25. Dynamic allocation of memory (with new)	3.97	1.02
26. Dynamic allocation of memory (with malloc)	4.03	1.01
Total Average score	4.00	1.02

5.3 Metacognitive Skills of Students for Computer Programming

In order to understand the effect of metacognitive skills towards problem solving in programming learning based on all dimensions, descriptive analysis was employed. From the result shown in Table 5.3, the total mean and standard deviation values for all 9 items were ($M = 1.24$; $SD = 0.49$). These results show that students have higher levels of metacognitive thinking by virtue of the nature of their specialization that needs planning, monitoring, and regulation. The highest opinion of students was obtained from item 6: “*When I program, I trace the program’s implementation with a trace table*” with ($M = 1.33$; $SD = 0.62$)” which is probably because the students who study programming languages have mathematical skills and they use them to easily track programs. The lowest opinion of students was obtained from item 1 “*I believe in planning, I always plan the results of my program to achieve the desired results* ($M = 1.17$; $SD = 0.45$)” which is probably because of the students’ lack of understanding on how the programs work and the program code.

Table 5.3: Mean and standard deviation for metacognitive skills of students for computer programming questionnaire

Planning	Mean	SD
1. I believe in planning, I always plan the results of my program to attain the desired results.	1.17	0.45
2. I always write down my plans to direct how I think when programming.	1.31	0.52
3. I always take time to think deeply the steps I should take to solve a new programming problem.	1.31	0.50
Monitoring		
4. During programming, I always pause in between coding sessions to check what I have already programmed.	1.19	0.42
5. I always ask myself questions just to ensure that I understand a challenging programming statement or question.	1.25	0.49
6. When I program, I trace the program's implementation with a trace table.	1.33	0.62
Regulation		
7. Even when the program is challenging to write, I go back and adjust it until the problem is solved effectively.	1.20	0.47
8. I take the programming reports that were done erroneously and change them until I have solved the problem well.	1.22	0.46
9. I read again the report of a challenging problem to ensure that I understood it in a proper way and that it is programmed correctly.	1.22	0.50
Total	1.24	0.49

5.3.1 Planning

In order to understand the effect of metacognitive skills towards problem solving in programming learning based on planning, descriptive analysis was employed. According to the results, the students gave very clear opinions based on their perspectives and practices in terms of metacognitive skills for computer programming over planning. From the results shown in Table 5.4, the total mean and standard deviation values for all 3 items are ($M = 1.26$; $SD = .49$). This study showed a highness amongst students in the planning skills meaning that the student had to plan from time to time in a comprehensive manner, in accordance with the circumstances and conditions during problem solving. In this stage the students always wrote down their plans as guidelines when programming. This finding was in line with the results of Al-Jarrah and Obeida (2011) in which they found high levels of

planning among university students. Nonetheless, these results differed with those of the study (Al-Saleem et al., 2012) which discovered that the levels of metacognitive thinking skills were at the moderate level in three dimensions. The highest opinion of students were obtained from items 2 and 3 “*I always write down strategies to direct my thinking when programming, I always take time to think deeply the steps I should take to solve a new programming problem*” with ($M= 1.31$; $SD=.52$), ($M = 1.31$; $SD = .50$)” respectively. The lowest opinion of students was calculated from item 1 “*I believe in planning, I always plan the solution of my program to attain the desired results*” ($M = 1.17$; $SD = .45$)”.

Table 5.4: Mean and standard deviations for every item of the planning dimension of the questionnaire

Planning	Mean	SD
1. I believe in planning, I always plan the solution of my program to achieve the desired results.	1.17	0.45
2. I always write down plans to direct my thinking when programming.	1.31	0.52
3. I always take time to think deeply the steps I should take to solve a new programming problem.	1.31	0.50
Total	1.26	0.49

5.3.2 Monitoring

In order to understand the effect of metacognitive skills towards problem solving in programming learning based on monitoring, descriptive analysis was employed. According to the results of “*Monitoring*”, the students gave very clear opinions based on their perspectives on their practice in terms of Metacognitive skills for computer programming over “*Monitoring*”. From the results shown in Table 5.5, the total mean and standard deviation values for all 3 items are ($M = 1.26$; $SD = 0.51$). This means that the level of monitoring skill in the study sample had the same planning skill value. The highest opinion of students were calculated from item 6 “*When I program, I trace the program’s implementation with a trace table*” ($M = 1.33$; $SD = 0.62$)”. The lowest opinion of students were calculated from item 4 “*During programming, I always pause in between coding sessions to check what I have already programmed*” ($M=1.19$; $SD=0.42$)”.

Table 5.5: Mean and standard deviation for every item of monitoring dimension of the questionnaire

Monitoring	Mean	SD
4. During programming, I always pause in between coding sessions to check what I have already programmed.	1.19	0.42
5. I always ask myself questions to ensure that I know a difficult programming statement or question.	1.25	0.49
6. When I program, I look at the program's execution with a trace table.	1.33	0.62
Total	1.26	0.51

5.3.3 Regulation

In order to understand the effect of metacognitive skills towards problem solving in programming learning based on regulation, descriptive analysis was employed. According to the results of regulation, the students gave very clear opinions based on their perspectives on their practices in terms of metacognitive skills for computer programming over-regulation. From the results shown in Table 5.6, The total mean and standard deviation values for all 3 items are ($M = 1.21$; $SD = .48$). This study showed a weakness among students in the skills of regulation. These results do not go in accordance with what has been obtained by (Aljaberi & Gheith, 2015) which reflects that there are weaknesses among learners in the regulation skills. The highest score on the opinion of students were calculated from items 8 and 9 “*I take the programming reports that have mistakes in them and change them until I have solved the problem in the correct way*” and “*I read again the description of a difficult problem to make sure that I fully understood it correctly and that it is programmed in the proper way* ($M=1.22$; $SD=.47$) , ($M = 1.22$; $SD = 0.46$)” respectively. The lowest score on the opinion of students was calculated from item 7 “*Even when the program is not easy to write, I go back and change it until the problem is solved properly* ($M = 1.20$; $SD = .47$)”.

Table 5.6: Mean and standard deviation for every item of regulation dimension of the questionnaire

Regulation	Mean	SD
7. Even when the program is not easy to write, I go back and change it until the problem is solved properly.	1.20	0.47
8. I take the programming statements that are done erroneously and change them until I have fixed the problem successfully.	1.22	0.47
9. I read again the description of a challenging problem to ensure that I understood it correctly and that it is programmed in a correct way.	1.22	0.46
Total	1.21	0.48

5.4 Problem Solving Skills of Students for Computer Programming

In order to understand the Problem-solving aids of students when learning computer programming language on all dimensions, descriptive analysis was employed. From the results shown in Table 5.7, the total mean and standard deviation values for all 32 items are ($M = 4.08$; $SD = 0.85$). The highest opinion of students were obtained from item 22 “*It is difficult to transform a textual problem into a mathematical formula that solves a given problem ($M = 4.19$; $SD = 0.82$)*” which is probably because most students have difficulties with mathematical problems solving. The lowest opinion of students were obtained from item 1 “*It is difficult to understand a program without first deriving its algorithm ($M = 3.62$; $SD = 1.02$)*” which is probably because many students make use of algorithms during programming.

Table 5.7: Mean and standard deviation for problem solving skills of students for computer programming of the questionnaire

System analysis	Mean	SD
1. It is difficult to understand a program without first deriving its algorithm.	3.62	1.02
2. It is difficult to understand a given programming problem	3.85	.89
3. It is difficult to interpret the problem.	3.98	.88
4. It is difficult to understand what is required for the program.	3.98	.85
5. It is difficult to write an algorithm for the problem.	4.04	.88
6. It is difficult to understand what is required by the programming problem.	4.07	.84
7. It is difficult to understand the logic of a large program.	3.88	.94
Design	M	SD
8. It is difficult to create suitable database.	4.04	.85
9. It is difficult to construct software models such as use-case diagrams.	4.01	.95
10. It is difficult to identify the important steps when solving a problem.	4.03	.89
11. It is difficult to identify possible methods.	4.09	.87
12. It is difficult to understand the logic of an algorithm.	4.12	.81
13. It is difficult to draw the flow chart of given programming task.	4.13	.85
14. It is difficult to write pseudocode of given programming task.	4.08	.83
15. It is difficult to model a software.	4.09	.88
16. It is difficult to do software documentation.	4.00	.86
17. It is difficult to simulate complex software.	4.07	.87
Coding	M	SD
18. It is difficult to connect a database to a programming language	4.10	.91
19. It is difficult to divide functionality into procedures.	4.11	.93
20. It is difficult to learn the programming language syntax.	4.05	.89
21. It is difficult to remember programming language syntax.	4.18	.87
22. It is difficult to develop a program that solves a given task.	4.17	.85
23. It is difficult to apply the correct logic.	4.15	.88
24. It is difficult to understand the concepts of programming structures.	4.15	.82
25. It is difficult to transform a textual problem into a mathematical formula that solves a given problem.	4.19	.82
26. It is difficult to make a combination of the necessary programming statements correctly in a program.	4.17	.83
Testing	M	SD
27. It is difficult to test programs especially detecting errors in the code.	4.06	.92
28. It is difficult to find bugs in my own programs.	4.25	.85
29. It is difficult to test the developed program using test data.	4.23	.83
30. It is difficult to write programs with no errors.	4.15	.87
31. It is difficult to test programs using functional test methods.	4.22	.85
32. It is difficult to test programs syntactically.	4.26	.82
Total	4.08	0.85

5.4.1 System Analysis

In order to understand the students' skills on problem-solving computer programming languages based on "system analysis", descriptive analysis was employed. According to the results of system analysis, the students gave very clear opinions based on their perspectives on their practices in terms of metacognitive skills for computer programming over system analysis. From the results shown in Table 5.8, the total mean and standard deviation values for all 9 items are ($M = 3.92$; $SD = 0.9$). The highest score on the opinion of students was obtained from item 6 "*It is difficult to understand what is required by the programming problem ($M = 4.07$; $SD = 0.84$)*" This shows that students were having problems in understanding what was required by the programming problem. The lowest score on the opinion of students was calculated from item 1 "*It is difficult to understand a program without first deriving its algorithm ($M = 3.62$; $SD = 1.02$)*"

Table 5.8: Mean and standard deviation for every item of system analysis dimension of the questionnaire

System analysis	Mean	SD
1. It is difficult to understand a program without first deriving its algorithm.	3.62	1.02
2. It is difficult to understand a given programming problem	3.85	.89
3. It is difficult to interpret the problem.	3.98	.88
4. It is difficult to understand what is required for the program.	3.98	.85
5. It is difficult to write an algorithm for the problem.	4.04	.88
6. It is difficult to understand what is required by the programming problem.	4.07	.84
7. It is difficult to understand the logic of a large program.	3.88	.94
Total	3.92	0.9

5.4.2 Design

In order to understand the students' skills on solving the problems related to computer programming languages based on design, descriptive analysis was employed. According to the results of "Design", the students gave very clear opinions based on their perspectives on their practices in terms of metacognitive skills for computer programming over "Design". From the results shown in Table 5.9, the total mean and standard deviation values for all 10 items are ($M = 4.07$; $SD = .87$). The highest score on the opinion of students were calculated from item 13 "*It is difficult to draw the flowchart of given programming task ($M = 4.13$; $SD = .85$)*" that means that the students "disagree difficult level" means that they were having problems while drawing the flowchart, and the lowest

opinion of students were calculated via items 16 “*It is difficult to do software documentation (M = 4.00; SD = .86)*” that means students “*disagree level*” is that that they were having problems while doing software documentation.

Table 5.9: Mean and standard deviation for each items of design dimension of the questionnaire

Design	Mean	SD
8. It is difficult to create suitable database.	4.04	.85
9. It is difficult to construct software models such as use-case diagrams.	4.01	.95
10. It is difficult to identify the important steps when solving a problem.	4.03	.89
11. It is difficult to identify possible methods.	4.09	.87
12. It is difficult to understand the logic of an algorithm.	4.12	.81
13. It is difficult to draw the flow chart of given programming task.	4.13	.85
14. It is difficult to write pseudocode of given programming task.	4.08	.83
15. It is difficult to model a software.	4.09	.88
16. It is difficult to do software documentation.	4.00	.86
17. It is difficult to simulate complex software.	4.07	.87
Total	4.07	.87

5.4.3 Coding

In order to understand the students’ skills on problem-solving computer programming languages based on coding, descriptive analysis was employed. According to the result of “*Coding*”, the students gave very clear opinions based on their perspectives on what they practice in terms of metacognitive skills for computer programming over “*Coding*”. From the result shown in Table 5.10, the total mean and standard deviation values for all 9 items are (M = 4.14; SD = .87). The highest opinion of students were calculated via items 25 “*It is difficult to transform a textual problem into a mathematical formula that solves a given problem (M = 4.19; SD = .82)*” that means students “*agree level*” means that they were having problems while transforming a textual problem into a mathematical formula that solves a given problem, and the lowest opinion of students were calculated via items 20 “*It is difficult to learn the programming language syntax (M = 4.05; SD = .89)*” that means students “*disagree level*” means that they were having problems while learning the programming language syntax.

Table 5.10: Mean and standard deviation for each items of coding dimension of the questionnaire

Coding	Mean	SD
18. It is difficult to connect a database to a programming language	4.10	.91
19. It is difficult to divide functionality into procedures.	4.11	.93
20. It is difficult to learn the programming language syntax.	4.05	.89
21. It is difficult to remember programming language syntax.	4.18	.87
22. It is difficult to develop a program that solves a given task.	4.17	.85
23. It is difficult to apply the correct logic.	4.15	.88
24. It is difficult to understand the concepts of programming structures.	4.15	.82
25. It is difficult to transform a textual problem into a mathematical formula that solves a given problem.	4.19	.82
26. It is difficult to make a combination of the necessary programming statements correctly in a program.	4.17	.83
Total	4.14	.87

5.4.4 Testing

In order to understand the students' skills on problem solving' computer programming languages based on testing, descriptive analysis was employed. According to the result of "Testing", the students gave very clear opinions based on their perspectives on what the practices in terms of metacognitive skills for computer programming over "Testing" were. From the result shown in Table 5.11, the total mean and standard deviation values for all 9 items are ($M = 4.20$; $SD = .86$). The highest opinion of students were calculated via items 32 "*It is difficult to test programs syntactically* ($M = 4.26$; $SD = .82$)" that means students "Strongly Disagree Level" means that they were having problems while testing programs syntactically, and the lowest opinion of students were calculated via items 27 "*It is difficult to test programs especially detecting errors in the code* ($M = 4.06$; $SD = .92$)" that means students "Disagree level" means that they were having problems while testing programs especially detecting errors in the code.

Table 5.11: Mean and standard deviation for each items of testing dimension of the questionnaire

Testing	Mean	SD
27. It is difficult to test programs especially detecting errors in the code.	4.06	.92
28. It is difficult to find bugs in my own programs.	4.25	.85
29. It is difficult to test the developed program using test data.	4.23	.83
30. It is difficult to write programs with no errors.	4.15	.87
31. It is difficult to test programs using functional test methods.	4.22	.85
32. It is difficult to test programs syntactically.	4.26	.82
Total	4.20	.86

5.5 Relationship between Sub-Dimensions of Metacognitive and Problem Solving Skills

For a better understanding of the relationship between sub-dimensions of metacognitive skills and problem solving skills, Pearson correlation analysis was also employed. As indicated in Table 5.12, in this study there is significant negative relationship between planning dimension and problem solving skills. In addition, there is significant negative relationship between monitoring dimension and problem solving skills. This therefore also indicated that there is significant negative relationship between regulation dimension and problem solving skills. For planning and problem solving skills, $r = -0.260^{**}$, $p = .000$, for monitoring and problem solving skills, $r = -0.265^{**}$, $p = .000$ and for regulation and problem solving skills $r = -0.262$, $p = .000$. A scatter plot below summarizes this finding:

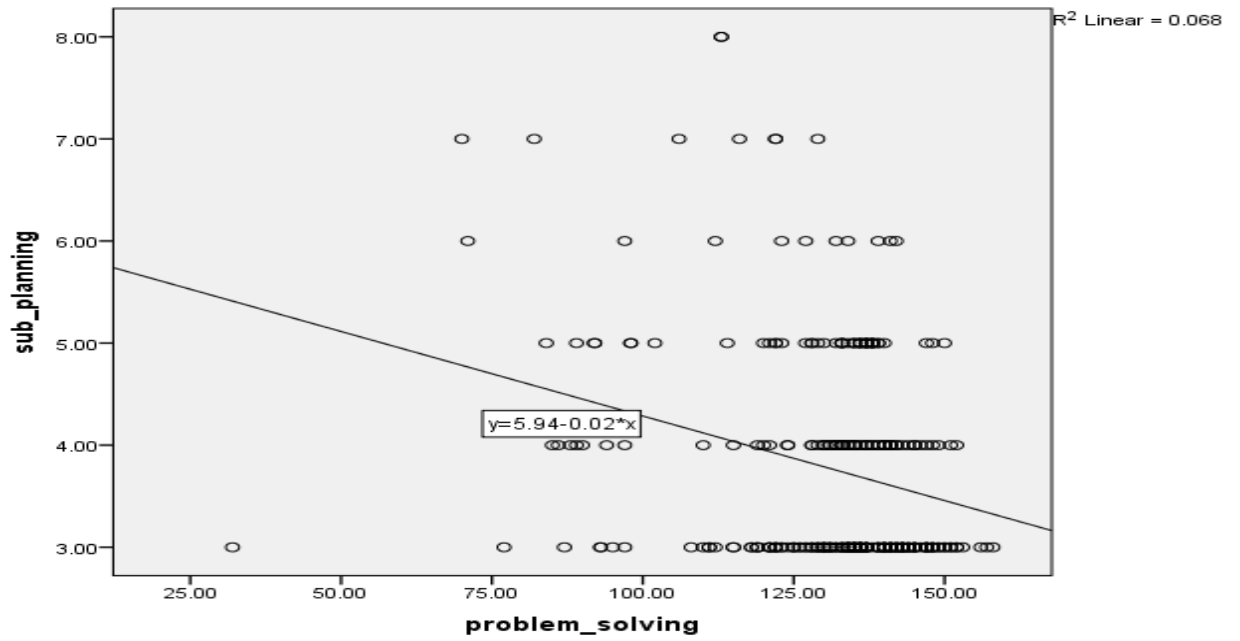


Figure 5.1: Scatter Plot between planning dimensions and problem solving skills

There exists a weak linear negative correlation between planning dimensions and problem solving skills.

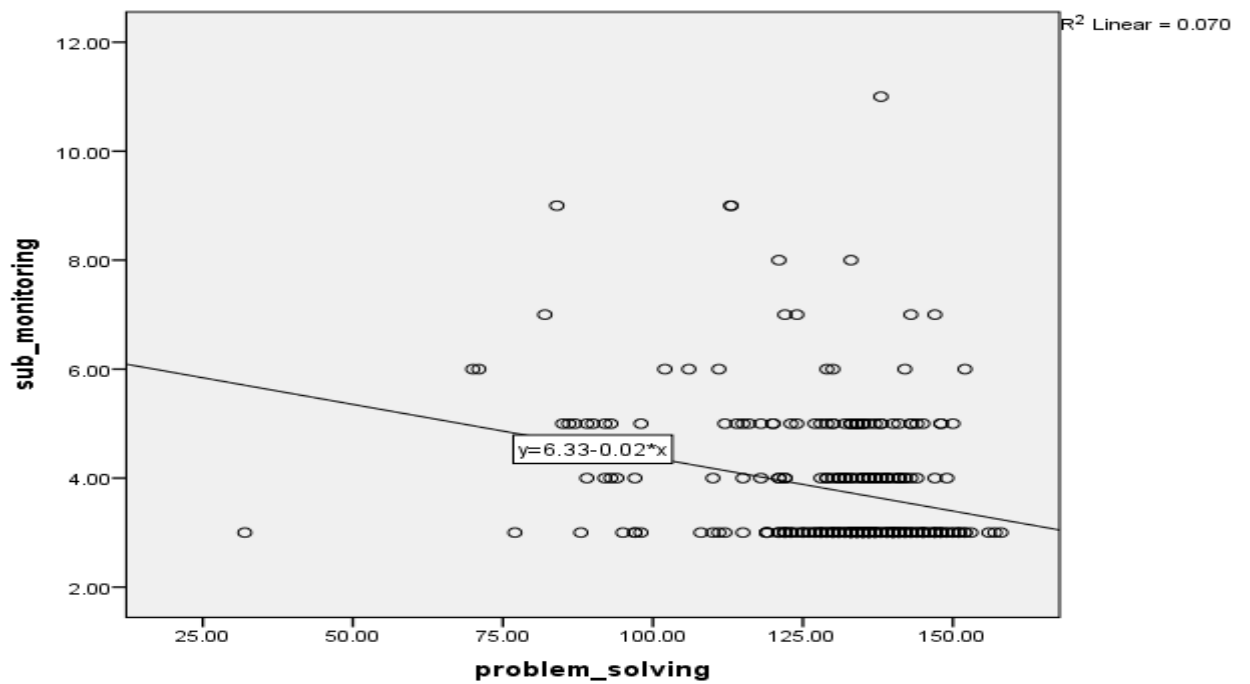


Figure 5.2: Scatter Plot between monitoring dimensions and problem solving skills

There exists a weak linear negative correlation between monitoring dimensions and problem solving skills.

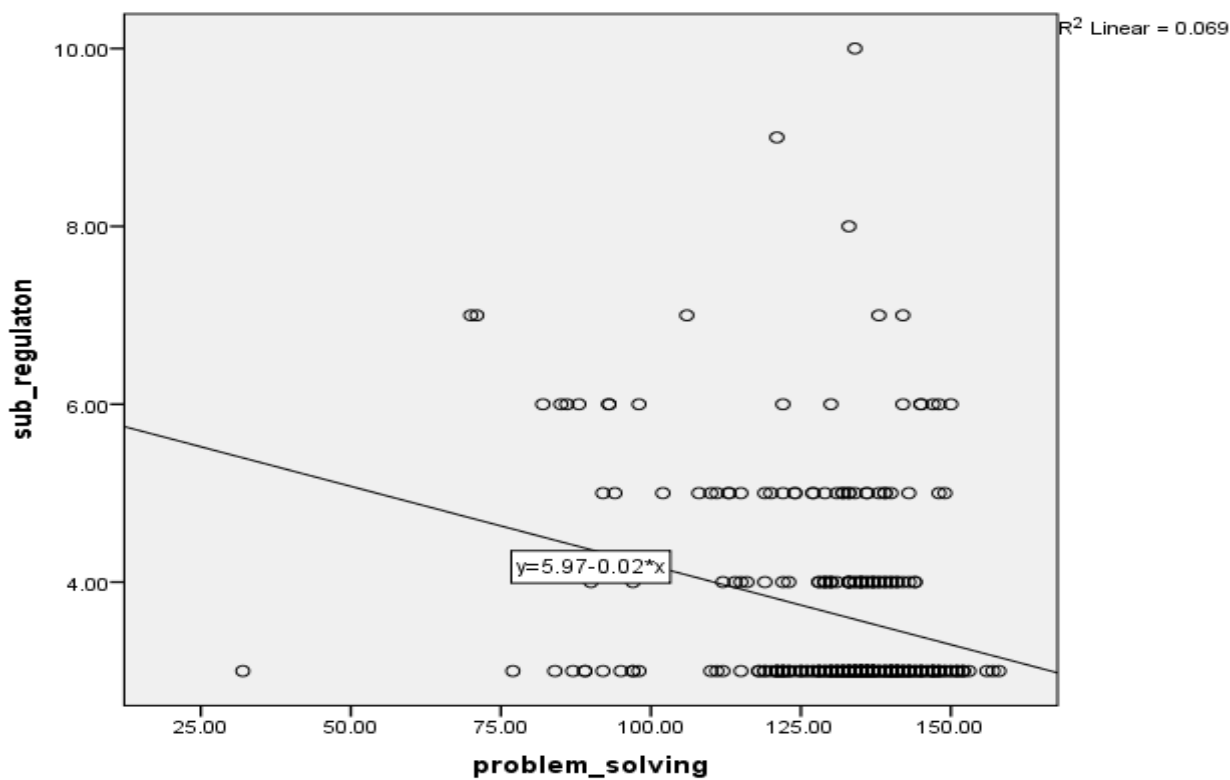


Figure 5.3: Scatter Plot between regulation dimensions and problem solving skills

There exists a weak linear negative correlation between regulation dimensions and problem solving skills.

Table 5.12: Relationship between sub-dimensions of metacognitive skills and problem solving skills

Correlation		
Sub-Dimensions	Statement	Problem Solving Skills
Planning	Person correlation	-.260 ^{**}
	Sig. (2-tailed)	0.000
	N	300
Monitoring	Person correlation	-.265 ^{**}
	Sig. (2-tailed)	0.000
	N	300
Regulation	Person correlation	-.262 ^{**}
	Sig. (2-tailed)	0.000
	N	300

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

5.6 Relationship between Metacognitive Skills and Problem Solving Skills

For a better understanding of the relationship between metacognitive skills and problem solving, Pearson correlation analysis was also employed. The results shown below in Table 5.13, shows that there is a negative relationship between metacognitive skills and problem-solving, the correlation coefficient is $-.321^{**}$ and its significance at the 0.01 significant level. There was a negative correlation between two variables, $r = -.321^{**}$, $n = 300$, $p = 0.000$. The negative correlation means that there is a negative relationship between the two variables, meaning that the increase in one of the variables is accompanied by a decrease in the second variable, and versa. A scatter plot in Figure 5.1 below summarizes this finding:

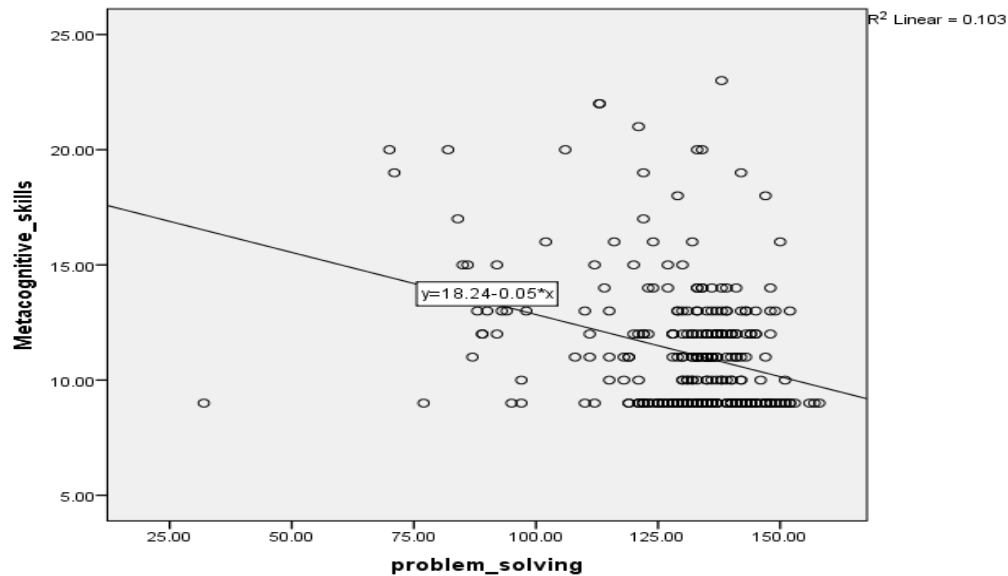


Figure 5.4: Scatter Plot between metacognitive skills and problem solving skills

There exists a weak linear negative correlation between metacognitive skills and problem solving skills.

Table 5.13: Relationship between metacognitive skills and problem solving skills

		Metacognitive skills	Problem solving
Metacognitive skills	Pearson	1	-.321
	Correlation		
	Sig. (2-tailed)		.000
	N	300	300
Problem solving	Pearson	-.321	1
	Correlation		
	Sig. (2-tailed)	.000	
	N	300	300

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

These results are not the same as the results of the studies carried out by (Ramesh & Anandraj, 2014; Kappa, 2001) where there is a major relationship between metacognition and problem solving. However these findings are consistent with (Coutinho, 2006) who

states that metacognitive thinking is not of help to students who are fulfilling the tasks of solving the problem, as students who had a advanced level of metacognitive thinking did not score any better as compared to the ones who had a lower level of metacognitive thinking. This result is not expected since it is rather expected that the way students perform in problem-solving should rise if the level of metacognitive rises as well, as those with higher levels of metacognitive thinking would be thought to make use of their metacognitive strategies in solving problems. This could be due to the problems that were too hard for students to solve, even if they knew metacognitive thinking strategies, but had not decided to use them in problem solving. This clearly reflects on the importance of using cognitive strategies in teaching, which shows an implication for further studies on the how important educational programs are and how to motivate students to improve and use these cognitive strategies in solving problems. (Pennequin et al., 2010) have put an emphasis on the importance of practice on using metacognitive thinking strategies in improving the students' abilities in solving problems, which is in line with (Coutinho, 2006; Abu- Alia & Alwaher, 2001; Al-Hamouri & Abu Mokh, 2011).

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

This chapter is a summary of the study and contains the conclusion as well as recommendations for future studies.

6.1 Conclusion

After conducting this study, it was discovered that metacognitive skills are indeed a successful learning style in enhancing student's logical thinking levels in programming classes. However, the results came in an unexpected way given the level of the relationship within the metacognitive skills and the capability to solving problems, evaluation has been conducted among 300 information science students and the results have:

- There is a negative correlation between the skills of sub-dimensions (planning, monitoring, regulation) and the ability to solving problems.
- Reported a negative correlation between students' metacognition and problem solving skills.
- The students seem not to know much about the script, and also they have average knowledge about the object orientated learning.
- The topic about memory allocation operations seems to be difficult for the students, and the topic about basic operations seems to be easy to learn.
- From the results, it can be inferred that the students' skills in problem solving skills towards computer programming languages based on system analysis, design and coding seems to be difficult which means that they have difficulties in issues related to making an algorithm to understanding a program, understanding and interpreting the problem and understanding and also interpreting design, while in testing students have fewer difficulties during the debugging of a program.
- This study reported that the level of university students' metacognitive skills was high in the monitoring and planning for problem solving in computer programming language.

6.2. Recommendations

Here is a list of suggestions, if enacted would add positively to improve the relationships within metacognition thinking and problem solving:

- Students must work hard to build a mental model to illustrate what is happening in the memory during program implementation
- Focus on practical lessons to help students learn programming languages
- Developing the student's capability to tackle mathematical and scientific problems and to increase students' understanding of these problems, and give these problem-solving skills in other to benefit student profit and alertness towards these skills.
- Students must learn how to understand programming language syntax, investigate programming problems and program debugging and the comprehension of error messages through intensive programming courses and collaborative learning that can be seen as a new way that can be integrated into programming classes in order to achieve better results in programming cycles.
- Allocation more time to the enhancement of metacognitive thinking skills is highly important and utilize these cognitive strategies in problem solving in all academic years and in all disciplines at university.

Hence, it is recommended to set up educational courses to enhance problem solving skills for students in North Cyprus and further work should look into this subject area.

Reference

- Abu-Alia, M., Alwaher, M.(2001). Degree of awareness of the Hashemite University students with knowledge of the cognitive skills related to preparation for exams, delivery and relationship with their level and on cumulative average and college to which they belong. *Amman-Jordan*, 28, 1-14.
- Agarwal, K. (2005). Python for CS1, CS2 and beyond. *Journal of Computing Sciences in Colleges*, 20(4), 262-270.
- Al-Jarrah, A., & Obeidat, A. (2011). Metacognitive thinking level amongst a sample of Yarmouk University students in the light of some variables. *Jordan Journal of Educational Sciences*, 7(2), 145-162.
- Aljaberi, N. M., & Gheith, E. (2015). University Students' Level of Metacognitive Thinking and their Ability to Solve Problems. *American International Journal of Contemporary Research*, 5(3), 121-134.
- Al-Saleem, B., Al-Rbabaah, J., & Al-Khawaldeh, K. (2012). The degree of acquiring metacognitive skills and its relationship with gender and specialisation and academic achievement in Jarash Secondary Schools. *The International Interdisciplinary Journal of Education*, 1(3), 73-87.
- Ali, M., Abd-Talib, C., Ibrahim, N. H., Surif, J., & Abdullah, A. H. (2016). The importance of monitoring skills in physics problem solving. *European Journal of Education Studies*.
- Alice (2015). An educational software that teaches students computer programming in 3D environment. Retrieved June 10, 2016 from <http://www.alice.org/index.php>
- Al-Hamouri, F., & Abu Mokh, A. (2011). Level of the need for cognition and metacognitive thinking among Yarmouk University Undergraduate Students, *Najah University Journal for Research*, 25(6), 1463-1488.
- Anandaraj, S., & Ramesh, C. (2014). A Study on the relationship between metacognition and problem solving ability of physics major students. *Indian Journal of Applied Research*, 4(5), 191-199.

- Annemieke, E., Jacobse, J., & Harskamp, E.G. (2012). Towards efficient measurement of metacognition in mathematical problem solving. *Springer*, 7, 133–149.
- Apiola, M., Tedre, M., & Oroma, J. O. (2011). Improving programming education in Tanzania: Teachers and students' perceptions. In *Proceedings of the 14th In Frontiers in Education Conference (FIE)*. (pp. 1-7). Tanzania: TUMAINI University.
- Arslan, S. (2014). An investigation of the relationships between metacognition and self-regulation with structural equation. *International Online Journal of Educational Sciences*, 6(3), 603-611.
- Arslan, S. & Akın, A. (2014). Metacognition: As a predictor of one's academic locus of control. *Educational Sciences: Theory & Practice*, 14(1), 1-8.
- Arslan, S., Akın, A. & Çitemel, N. (2013). The predictive role of grit on metacognition in Turkish university students. *Studia Psychologica*, 55(4). 311-320.
- Aurah, C. M., Cassady, J. C., & McConnell, T. J. (2014). Predicting problem solving ability from metacognition and self-efficacy beliefs on a cross validated sample. *British Journal of Education*, 2(1), 49-72.
- Avargil, S., Lavi, R., & Dori, Y. J. (2018). Students' metacognition and metacognitive strategies in science education. In *Cognition, Metacognition, and Culture in STEM Education*, (pp. 33-64). Springer, Cham.
- Baars, M., Van Gog, T., de Bruin, A., & Paas, F. (2017). Effects of problem solving after worked example study on secondary school children's monitoring accuracy. *Educational Psychology*, 37(7), 810-834.
- Barnes, D. J., Kölling, M., & Gosling, J. (2017). Objects first with Java: A practical introduction using BlueJ. *Pearson*.
- Biju, S.M. (2013). Difficulties in understanding object oriented programming concepts. Retrieved August 28, 2017 from <http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1440&context=dubaipapers>
- Bouvier, D., Lovellette, E., Matta, J., Alshaigy, B., Becker, B. A., Craig, M., & Zarb, M. (2016). Novice Programmers and the Problem Description Effect. In *Proceedings of the 2016 ITiCSE Working Group Reports* (pp. 103-118). ACM.

- Butler, M., & Morgan, M. (2007). Learning challenges faced by novice programming students studying high level and low feedback concepts. *In Proceedings of ascilite Singapore* (pp. 99-107). Narre Warren: Monash University.
- Carlisle, M.C. (2009). Raptor: A visual programming environment for teaching object-oriented programming. Retrieved November 9, 2016 from http://www.martincarlisle.com/publications/ccsc_named.pdf
- Chetty, J. & Westhuizen, D. (2014). Implementing metacognition skills for learners studying computer programming. *In Proceedings of EdMedia: World Conference on Educational Media and Technology, 1*, 726-731. Retrieved June 12, 2016 from <https://www.learntechlib.org/p/147775>
- Churchill, E. F., Bowser, A., & Preece, J. (2013). Teaching and learning human-computer interaction: past, present, and future. *interactions*, 20(2), 44-53.
- Costa, J. M., & Miranda, G. L. (2017). Relation between Alice software and programming learning: A systematic review of the literature and meta-analysis. *British Journal of Educational Technology*, 48(6), 1464-1474.
- Costa, L., and Kallick, B. (2001). What are Habits of Mind? Retrieved November 10, 2017 from <http://www.habits-of-mind.net/whatare.htm>
- Coutinho, S. A. (2006). The relationship between the need for cognition, metacognition, and intellectual task performance. *Educational Research and Reviews*, 1(5), 162-164.
- Crowfoot, S. (2012). Five computer programs we use every day. Retrieved September 28, 2017 from <http://www.iceni.com/blog/five-computer-programs-we-use-every-day/>
- Donchev, I., & Todorova, E. (2008). Object-oriented programming in Bulgarian universities informatics and computer science curricula. *Journal of Informatics and Education*, 7(2), 159-172.
- Ellis, S. M., & Steyn, H. S. (2003). Practical significance (effect sizes) versus or in combination with statistical significance (p-values): Research note. *Management dynamics: journal of the southern african institute for management scientists*, 12(4), 51-53.

- Erol, O., & Kurt, A. A. (2017). The effects of teaching programming with scratch on pre-service information technology teachers' motivation and achievement. *Computers in Human Behavior*, 77, 11-18.
- Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive developmental inquiry. *American Psychologist*, 34(10), 906-911.
- Fu, X., Shimada, A., Ogata, H., Taniguchi, Y., & Suehiro, D. (2017). Real-time learning analytics for C programming language courses. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference* (pp. 280-288). ACM.
- Garrison, D.R., & Akyol, Z. (2015). Toward the development of a metacognition construct for communities of inquiry. *Internet and Higher Education*, 24, 66-71.
- Geiwitz, J (1994). Training metacognitive skills for problem solving. *U.S. Army research institute for the behavioural and social sciences*. Retrieved December 27, 2016 from http://www.au.af.mil/au/awc/awcgate/army/ari_rn-1995-03.pdf
- Ginat, D., & Shmallo, R. (2013). Constructive use of errors in teaching. *SIGCSE Journal*, 1(2), 15-24.
- Gomes, A., & Mendes, A. J. (2007). Learning to program - difficulties and solutions. In *Proceedings of the 3th International Conference on Engineering Education – ICEE* (Vol 2, 3-7). Coimbra, Portugal.
- Harandi, V., Eslami Sharbabaki H., Ahmadi Deh M., & Darehkordi A. (2013). The effect of metacognitive strategy training on social skills and problem solving performance. *Journal of Psychology & Psychotherapy*, 3(4), 121-125.
- Harrykar. (2015). Running with Raptor. Retrieved November 22, 2017 from <http://progr-harrykar.blogspot.com.cy/2015/08/running-with-raptor.html>
- Havenga, M. (2011). Problem-solving processes in computer programming: a case study. In *Southern African Computer Lecturers' Association (SACLA) Conference Proceedings* (pp. 91-99).

- Hong, T. Y., & Chu, H. C. (2017). Effects of a Situated 3D Computational Problem-Solving and Programming Game-Based Learning Model on Students. *In 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)* (pp. 596-600). IEEE.
- Hu, M., Winikoff, M., & Cranefield, S. (2012). Teaching novice programming using goals and plans in a visual notation. *In Proceedings of the Fourteenth Australasian Computing Education Conference* (Vol 12, pp. 43-52). Australian Computer Society, Inc.
- Kafadar, H. (2012). Cognitive Model of Problem Solving. *In Yeni Symposium* (Vol. 50, No. 4)
- Kasemsap, K. (2017). Advocating problem-based learning and creative problem-solving skills in global education. *Handbook of research on creative problem-solving skill development in higher education*, 351-377.
- Kirsti, A. (2004). Problems in learning and teaching programming: A literature study for developing visualizations in the Codewitz-Minerva project. Retrieved on April 9, 2017 from https://www.cs.tut.fi/~edge/literature_study.pdf
- Kunkle, W. (2010). The impact of different teaching approaches and languages on student learning of introductory programming concepts. Retrieved June 9 2016 from <http://dl.acm.org/citation.cfm?id=2785807>
- Kunkle, W. M., & Allen, R. B. (2016). The impact of different teaching approaches and languages on student learning of introductory programming concepts. *ACM Transactions on Computing Education (TOCE)*, 16(1), 3.
- Luxton-Reilly, A., & Petersen, A. (2017). The Compound Nature of Novice Programming Assessments. *In Proceedings of the Nineteenth Australasian Computing Education Conference* (pp. 26-35). ACM.
- Madden, M., & Chambers, D. (2002). 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, 2002* (pp. 125-130). National University of Ireland.

- Maheswari, T. L., Ahamed, M. S., & Duraisamy, S. (2017). Quality assessment system for Object Oriented structure. *Journal of Computational and Theoretical Nanoscience*, 14(4), 1993-2014.
- Mahmoud, M., Mhashi, A., & Ali, A. (2013). Difficulties Facing Students in Learning Computer Programming Skills at Tabuk University. *In Proceedings of the 12th International Conference on Education and Educational Technology* (vol 5, pp. 15-24). Iwate: Japan.
- Matsuzawa, Y., Tohyama, S., & Sakai, S. (2014). A course design to develop meta-cognitive skills for collaborative knowledge building through tool-assisted discourse analysis. *International Journal of Organisational Design and Engineering*, 3(3-4), 260-277.
- Montague, M., Krawec, J., Enders, C., & Dietz, S. (2014). The effects of cognitive strategy instruction on math problem solving of middle-school students of varying ability. *Journal of Educational Psychology*, 106(2), 469.
- Mokos, E., & Kafoussi, S. (2013). Elementary students' spontaneous metacognitive functions in different types of mathematical problems. *Journal of Research in Mathematics Education*, 2(2), 242-267.
- Moström, J. E. (2011). A Study of student problems in learning to program. Department of Computing Science Umeå University, SE-901 87 Umeå, Sweden. Retrieved November 9, 2016 from www.cs.umu.se
- Myers, B. A., Stefik, A., Hanenberg, S., Kaijanaho, A. J., Burnett, M., Turbak, F., & Wadler, P. (2016). Usability of Programming Languages: Special Interest Group (SIG) Meeting at CHI 2016. *In Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 1104-1107). ACM.
- Özgür, Z. (2017). Introduction to computer programming. Retrieved February 16, 2017 from <http://landofcode.com/programming-intro/computer-programming-languages.php>
- Özsoy, G., & Ataman, A. (2017). The effect of metacognitive strategy training on mathematical problem solving achievement. *International Electronic Journal of Elementary Education*, 1(2), 67-82.

- Pennequin, V., Sorel, O., & Mainguy, M. (2010). Metacognition, executive functions and aging: the effect of training in the use of metacognitive skills to solve mathematical word problems. *Journal of Adult Development*, 17(3), 168-176.
- Ritchie, J., Lewis, J., Nicholls, C. M., & Ormston, R. (Eds.). (2013). *Qualitative research practice: A guide for social science students and researchers*. Sage.
- Rizk, N. M. H., Attia, K. A. M., & Al-Jundi, A. A. H. (2017). The Impact of Metacognition Strategies in Teaching Mathematics among Innovative Thinking Students in Primary School, Rafha, KSA. *International Journal of English Linguistics*, 7(3), 103.
- Rum, S. N. M., & Ismail, M. A. (2014). Usability evaluation of metacognitive support system for novice programmers. *Asian Journal of Education and e-Learning*, 2(5), 2321 – 2454.
- Safari, Y. & Meskini, H. (2015). The effect of metacognitive instruction on problem solving skills in Iranian students of health sciences. *Global Journal of Health Science*, 8(1), 150-156.
- Sajaniemi, J., Kuittinen, M., & Tikansalo, T. (2008). A study of the development of students' visualizations of program state during an elementary object-oriented programming course. *Journal on Educational Resources in Computing*, 7(4), 3.
- Siegfried, R. M., Siegfried, J., & Alexandro, G. (2016). A Longitudinal Analysis of the Reid List of First Programming Languages. *Information Systems Education Journal*, 14(6), 47.
- Sipahi, B., & Yurtkoru, E. S. ve Çinko, M. (2010). *Sosyal bilimlerde SPSS'le veri analizi*. Istanbul: Bete Yayinlari.
- Siswati, B. H., & Corebima, A. D. (2017). The Effect of Education Level and Gender on Students' Metacognitive Skills in Malang, Indonesia. *Advances in Social Sciences Research Journal*, 4(4).
- Stevenson, J., & Wood, M. (2017). Recognising object-oriented software design quality: a practitioner-based questionnaire survey. *Software Quality Journal*, 1-45.

- Tas, C., Brown, E. C., Aydemir, O., Brüne, M., & Lysaker, P. H. (2014). Metacognition in psychosis: Comparison of schizophrenia with bipolar disorder. *Psychiatry Research*, 219(3), 464-469.
- Titus, S.V., & Annaraja, P. (2011). Teaching competency of secondary teacher education students in relation to their metacognition. *International Journal on New Trends in Education and their Implications*, 2(3), 14-22.
- Valdecantos, H. A., Tarrit, K., Mirakhorli, M., & Coplien, J. O. (2017). An empirical study on code comprehension: data context interaction compared to classical object oriented. In *Proceedings of the 25th International Conference on Program Comprehension* (pp. 275-285). IEEE Press.
- Vasconcelos, C., & Ravara, A. (2017). From object-oriented code with assertions to behavioural types. In *Proceedings of the Symposium on Applied Computing* (pp. 1492-1497). ACM.
- Wang, X. M., & Hwang, G. J. (2017). A problem posing-based practicing strategy for facilitating students' computer programming skills in the team-based learning mode. *Educational Technology Research and Development*, 65(6), 1655-1671.
- Xinogalos, S., Satratzemi, M & Dagdilelis, V. (2006). Evaluating object Karel - an educational programming environment for object oriented programming. *Current Developments in Technology-Assisted Education*, 2, 148-171.
- Young, A., & Fry, J. (2012). Metacognitive awareness and academic achievement in college students. *Journal of the Scholarship of Teaching and Learning*, 8(2), 1-10.

APPENDICES

APPENDIX A
RAPPORTEUR OF THE SCIENTIFIC RESEARCH ETHICS COMMITTEE

Dear Raja Jwadi H. Fourti,

11.09.2017

Your application titled **“The Effect of Metacognitive Skills Towards Problem Solving In Programming Learning”** with the application number YDÜ/FB/2017/8 has been evaluated by the Scientific Research Ethics Committee and granted approval. You can start your research on the condition that you will abide by the information provided in your application form.

Assist. Prof. Dr. Direnç Kanol

Rapporteur of the Scientific Research Ethics Committee



Note: If you need to provide an official letter to an institution with the signature of the Head of NEU Scientific Research Ethics Committee, please apply to the secretariat of the ethics committee by showing this document.

UNIVERSITY STUDENTS' METACOGNITIVE AND PROBLEM SOLVING SKILLS TOWARDS LEARNING A PROGRAMMING LANGUAGE

Thank you in advance for taking time to answer our survey.

1. **Gender:** a) Male b) Female
2. **Age:** a) 18-20 b) 21-24 c) 25+
3. **Nationality:** a) Cypriot b) Turkish c) Nigerian d) Libyan e) Iraqi
f) Other
4. **Your Native Language:** a) Turkish b) English c) Arabic d) Others
5. **Department:** a) Computer Information Systems b) Computer Engineering
c) Information technology d) Management Information Systems
6. **GCPA:** a) 2.00 – 2.99 b) 3.00 – 3.49 c) 3.50 – 4.00
8. **What is your level of study?** a) Undergraduate b) Master c) PhD
9. **Which operating system do you use?** a) Windows b) Linux/Unix c) Macintosh
10. **University name:** _____

11. Which programming languages do you know?

	Very Good	Above Average	Average	Little	Don't Know
OBJECT ORIENTATED					
C++					
C#					
Visual basic					
Java					
.NET					
WEB BASED					
ASP					
HTML					
JavaScript					
PHP					
SCRIPT BASED					
C language					
Pascal					
Pear					
Fortran					
Python					
Delphi					
Ruby					

12. How difficult do you believe each of the following topics are to learn?

	Very Easy	Easy	Moderate	Difficult	Very Difficult
BASIC OPERATIONS					
1. Variable/function declarations					
2. Mathematical operators and precedence					
3. Conditional operations (if, else, etc.)					
4. Looping operations (for, while, etc.)					
5. Input/output and file handling					
MORE COMPLEX OPERATIONS					
6. Arrays					
7. Other data structures (trees, linked-lists)					
8. String handling					
9. Structures					
10. Pointers					
11. Recursion					
OBJECT ORIENTED OPERATIONS					
12. Classes and objects					
13. Casting					

14.	Inheritance					
15.	Encapsulation					
16.	Polymorphism					
17.	Passing by reference/passing by value					
18.	Function overloading/default arguments					
19.	Function over-riding (in inheritance)					
20.	Constructors/destructors					
21.	Templates					
22.	Copy constructors					
23.	Operator overloading					
24.	Virtual functions					
MEMORY ALLOCATION OPERATIONS						
25.	Dynamic allocation of memory (with new)					
26.	Dynamic allocation of memory (with mallow)					

SECTION II: Metacognitive skills for computer programming

	Always	Often	Seldom	Never
Planning				
1. I believe in planning, I always plan the solution of my program to achieve the desired results.				
2. I always write down plans to direct my thinking when programming.				
3. I always take time to think deeply the steps I should take to solve a new programming problem.				
Monitoring				
4. During programming, I always pause in between coding sessions to check what I have already programmed.				
5. I always ask myself questions to make sure that I understand a difficult programming statement or question.				
6. When I program, I trace the program's execution with a trace table.				
Regulation				
7. Even when the program is difficult to write, I go back and modify it until the problem is solved successfully.				
8. I take the programming statements that have errors in them and adjust them until I have solved the problem successfully.				
9. I reread the description of a difficult problem to make sure that I understood it correctly and that it is correctly programmed.				

SECTION III: Problem solving skills (Please tick the most appropriate ones to you)

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
SYSTEM ANALYSIS					
1. It is difficult to understand a program without first deriving its algorithm.					
2. It is difficult to understand a given programming problem.					
3. It is difficult to interpret the problem.					
4. It is difficult to understand what is required for the program.					
5. It is difficult to write an algorithm for the problem.					
6. It is difficult to understand what is required by the programming problem.					
7. It is difficult to understand the logic of a large program.					
DESIGN					
8. It is difficult to create suitable database.					
9. It is difficult to construct software models such as use-case diagrams.					
10. It is difficult to identify the important steps when solving a problem.					
11. It is difficult to identify possible methods.					
12. It is difficult to understand the logic of an algorithm.					
13. It is difficult to draw the flow chart of given programming task.					
14. It is difficult to write pseudocode of given programming task.					
15. It is difficult to model a software.					
16. It is difficult to do software documentation.					
17. It is difficult to simulate complex software.					
CODING					
18. It is difficult to connect a database to a programming language					
19. It is difficult to divide functionality into procedures.					
20. It is difficult to learn the programming language syntax.					
21. It is difficult to remember programming language syntax.					
22. It is difficult to develop a program that solves a given task.					
23. It is difficult to understand the concepts of programming structures.					
24. It is difficult to apply the correct logic.					
25. It is difficult to transform a textual problem into a mathematical formula that solves a given problem.					
26. It is difficult to combine the necessary programming statements successfully in a program.					
TESTING					
27. It is difficult to test programs especially detecting errors in the code.					
28. It is difficult to find bugs in my own programs.					
29. It is difficult to test the developed program using test data.					
30. It is difficult to write programs with no errors.					
31. It is difficult to test programs using functional test methods.					
32. It is difficult to test programs syntactically.					

Thank you for taking time to participate in the survey ☺ ☺ ☺

APPENDIX C SIMILARITY REPORT

← → 🔒 Secure | https://turnitin.com/t_inbox.asp?aid=64670438&lang=en_us&session-id=94bf4eba519a8361b2393365236be065# ☆

been generated.

TEZ

INBOX | NOW VIEWING: NEW PAPERS ▼

Submit File
Online Grading Report | Edit assignment settings | Email non-submitters

<input type="checkbox"/>	AUTHOR	TITLE	SIMILARITY	GRADE	RESPONSE	FILE	PAPER ID	DATE
<input type="checkbox"/>	Rajaa Aljwadi	chp6	0% <div style="width: 0%; height: 10px; background-color: #007bff;"></div>	--	--		900920250	08-Jan-2018
<input type="checkbox"/>	Rajaa Aljwadi	Chp1	2% <div style="width: 2%; height: 10px; background-color: #28a745;"></div>	--	--		900919034	08-Jan-2018
<input type="checkbox"/>	Rajaa Aljwadi	Abstract	3% <div style="width: 3%; height: 10px; background-color: #28a745;"></div>	--	--		900918914	08-Jan-2018
<input type="checkbox"/>	Rajaa Aljwadi	chp3	6% <div style="width: 6%; height: 10px; background-color: #28a745;"></div>	--	--		900919770	08-Jan-2018
<input type="checkbox"/>	Rajaa Aljwadi	chp4	7% <div style="width: 7%; height: 10px; background-color: #28a745;"></div>	--	--		900919908	08-Jan-2018
<input type="checkbox"/>	Rajaa Aljwadi	All_Thesis	8% <div style="width: 8%; height: 10px; background-color: #28a745;"></div>	--	--		900920713	08-Jan-2018
<input type="checkbox"/>	Rajaa Aljwadi	chp5	8% <div style="width: 8%; height: 10px; background-color: #28a745;"></div>	--	--		900920159	08-Jan-2018
<input type="checkbox"/>	Rajaa Aljwadi	chp2	15% <div style="width: 15%; height: 10px; background-color: #28a745;"></div>	--	--		900919399	08-Jan-2018

Copyright © 1998 – 2018 Turnitin, LLC. All rights reserved.

[Privacy Policy](#)
[Privacy Pledge](#)
[Terms of Service](#)
[EU Data Protection Compliance](#)
[Copyright Protection](#)
[Legal FAQs](#)
[Helpdesk](#)
[Research Resources](#)

TUR 15:36
8.01.2018