A METACOGNITIVE SUPPORT ENVIRONMENT FOR NOVICE PROGRAMMER USING SEMANTIC WEB

SITI NURULAIN BT MOHD RUM

THESIS SUBMITTED IN FULFILLMENT OF THE REQUIREMENTSFOR THE DEGREE OF DOCTOR OF PHILOSOPHY

**FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITI OF MALAYA
KUALA LUMPUR**

2015

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **SITI NURULAIN BINTI MOHD**

Registration/Matric No: **WHA120021**

Name of Degree: **DOCTOR OF PHILOSOPHY**

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

A METACOGNITIVE SUPPORT ENVIRONMENT FOR NOVICE PROGRAMMER USING SEMANTIC WEB

Field of Study: **COMPUTER SCIENCE (INFORMATION SYSTEM)**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:
Designation

ABSTRACT

It is has been acknowledged that acquiring programming knowledge is very challenging for someone that is very new to programming. Computer programming education for both teaching and learning is generally known to be difficult as it requires a higher level of knowledge and practice ability rather than theory alone. The skill of metacognition, which is apparently lacking among novices is required for the knowledge of “When and why” in computer programming practice to manage cognitive skills efficiently and facilitate the learning process as well as the transfer of knowledge. This study aims to identify the methods of supporting novice programmers in learning computer programming metacognitively. Several steps are taken in realizing this. Firstly, a survey using Metacognitive Awareness Inventory (MAI) is performed to investigate the effect of metacognition on the performance in respect of the Computer Programming subject. The respondents that participated in this study are the undergraduate students with a computer science background from several universities in Malaysia. The result indicates that metacognition has a positive effect on students’ achievement in the Computer Programming Subject at university. The results also revealed that i) problem-solving is the type of strategy that usually works for them to understand programming, ii) novices need a support environment to assist them to monitor and evaluate their knowledge in learning programming, and iii) the majority of the novices are motived to use a support environment if it provides criteria like learnability, helpfulness and affect. Secondly, interview sessions are conducted with expert lectures in Computer Programming at the Faculty of Computer and Mathematical Science, University Teknologi MARA, Shah Alam. The objective of the interview was to investigate the awareness and the implementation of metacognition in teaching Introductory Computer Programming at the University. The findings of the survey show that i) expert lecturers employed metacognition in teaching, and teaching metacognition skills for improving student’s learning process and ii) expert lecturers recognize the role of metacognition as one of an important learning success factors of the computer programming subject that must be done before, during and after instruction. Semantic Web is an ideal technology for the development of the proposed system that facilitates the process of differentiating the ambiguity lies between cognitive and metacognitive elements. Thirdly, a usability test is conducted using the Software Usability Measurement Inventory (SUMI) to determine the user’s perception towards the MSSNP in terms of the affect, efficiency, control, helpfulness and learn-ability. The results of the usability test show that the MSSNP is an ideal system to support the metacognitive activities for learning Introductory Computer Programming. Finally, an experimental study is conducted, which comprise an empirical evaluation of observations on student interaction with the MSSNP. It concentrates primarily on the observation of metacognitive changes as well as changes in performance. The subjects are categorized into two groups (experimental and control) in which each one are associated with different conditions. The results show that students in the experimental group developed more favorably compared to those in the control group with respect to metacognition behavior and performance.

ABSTRAK

Untuk memperolehi pengetahuan dalam bidang pengaturcaraan Komputer telah diketahui umum sebagai sesuatu yang amat mencabar untuk seseorang yang baru dalam dunia pengaturcaraan. Pengajaran dan pembelajaran Pengaturcaraan Komputer telah diketahui umum sebagai sesuatu yang sukar dan memerlukan keupayaan ilmu pengetahuan yang tinggi. Ini merujuk kepada pengetahuan "Bila dan Mengapa" yang memerlukan kemahiran-kemahiran metakognitif yang tidak dipunyai oleh kebanyakkan Pengaturcara Baru. Metakognitif merupakan satu kemahiran penting kepada Pengaturcara Baru bagi menguruskan kemahiran kognitif mereka dengan cekap dan menjadikan proses pembelajaran lebih mudah yang secara tidak langsung dapat memudahkan proses pemindahan pengetahuan. Tujuan kajian ini dijalankan adalah untuk mengenalpasti kaedah-kadeah yang boleh menyokong Pengaturcara Baru dalam pembelajaran Pengaturcaraan Komputer secara metakognitif. Beberapa langkah diambil dalam merealisasikan kajian ini. Pertama sekali, kajian menggunakan ‘Metacognitive Awareness Inventory’ (MAI) telah dijalankan bagi megenalpasti kepentingan metakognisi dalam pencapaian prestasi pembelajaran Pengaturcaraan Komputer di Universiti. Responden yang menyertai kajian ini adalah pelajar mahasiswa yang mempunyai latar belakang Sains Komputer dari beberapa universiti di Malaysia. Hasil Kajian telah menunjukkan bahawa metakognisi mempunyai kesan yang positif dalam pencapaian pelajar dalam pembelajaran Pengaturcaraan Komputer di universiti. Keputusan juga menunjukkan i) ‘Penyelesaian masalah’ adalah strategi yang lazimnya digunakan oleh pelajar bagi mefahami pembelajaran pengaturcaraan, ii) Pengaturcara Baru memerlukan persekitaran sokongan yang boleh membantu mereka dalam memantau dan menilai ilmu pengetahuan mereka iii) majoriti Pengaturcara Baru mempunyai motivasi kepada persekitaran sokongan sekiranya Kriteria seperti ‘Keupayaan belajar’, ‘Kemudahan bantuan’ dan serta ‘memberi kesan’ diambilkira .

Kedua, sesi temu duga telah dibuat bersama pakar pengajar kursus Pengenalan Pengaturcaraan Komputer yang di Fakulti Komputer & Sains Matematik di UiTM ,Shah Alam. Objektif temuduga adalah untuk mengenalpasti sejauh manakah kesedaran dan pelaksanaan metakognisi di dalam pengajaran Pengenalan Pengaturcaraan Komputer di Universiti. Penemuan kajian telah menunjukkan i) Pensyarah pakar menggunakan metakognisi di dalam pengajaran, serta mengajar kemahiran-kemahiran metacognition kepada pelajar dalam meningkatkan pengajian mereka di perkara pengaturcaraan computer dan ii) pensyarah-pensyarah pakar menyedari akan peranan metakognisi sebagai salah satu faktor penting kejayaan dalam pengaturcaraan komputer yang perlu diaplikasikan sebelum, semasa dan selepas pengajaran. Teknologi Semantic Web sangat ideal bagi membangunkan sistem yang dicadangkan bagi memudahkan proses menyelesaikan masalah kekaburan elemen di antara kognitif dan metakognitif. Ketiga, satu ujian ‘kebolehgunaan’ dijalankan menggunakan Software Usability Measurement Inventory (SUMI) bagi melihat penerimaan pengguna terhadap MSSNP dari segi kesan, kecekapan, kawalan, bantuan dan kebolehan menelaah. Keputusan ujian ‘kebolehgunaan’ telah menunjukkan MSSNP merupakan satu sistem yang dapat membantu dalam menyokong aktiviti-aktiviti metakognitif dalam pembelajar Pengaturcaraan Komputer. Akhir sekali,satu eksperimen telah dijalankan secara empirikal bagi melihat kesan interaksi pelajar terhadap MSSNP. Tumpuan diberikan kepada pemerhatian metakognitif dan perubahan prestasi dalam pembelajaran. Ekseperimen kajian telah dibahagikan kepada dua kumpulan eksperimen dan kumpulan yang dikawal. Pelajar dalam kumpulan eksperimen telah menunjukkan perubahan yang baik didalam prestasi pembelajaran dan perlakuan metakognitif.

ACKNOWLEDGEMENTS

First of all, praise be to Allah the Almighty for his compassion and mercifulness in allowing me to finalize this research. There are various people who I would like to thank, without whom this thesis would not have been possible. First of all, to my supervisor, Dr. Maizatul Akmar Ismail, for her continuous support of this research work; for her enthusiasm, motivation, patience and immense knowledge in supervising me. Her guidance helped me throughout the research and writing of this thesis. I would also like to express my thanks to the Doctorate Support group (DSG), which provided many helpful tips on how to conduct research. Last, but not least, I would like to thank my family, especially to my husband and my parents for their continuous support and prayers for me for completing this study successfully. It would not have been possible without their prayers, encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GPA	Grade Point Average
CDL	Conditional
CM	Comprehension Monitoring
DL	Declarative
DVG	Debugging Strategies
EVL	Evaluation
ILE	Instructional Learning Environment
IMS	Information Management Strategies
ITS	Intelligent Tutoring System
KC	Knowledge of Cognition
KM	Knowledge Monitoring
KMA	Knowledge Monitoring Assessment
MAI	Metacognitive Awareness Inventory
OWL	Ontology Web Language
P	Procedural
PL	Planning
RC	Regulation of Cognition
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SRL	Self-Regulated Learning
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XML	Extensible Markup Language

CHAPTER 1: INTRODUCTION

It has been noted that developing programming skills is challenging for novice programmers (Falkner & Palmer, 2009; Soloway & Spohrer, 2013) and this is a Universal Problem. The research by (Ismail, Ngah, & Umar, 2010; Mason, 2012) indicated that learning to understand and code computer programs is extensively perceived to be very challenging in computer science education due to the demand for a greater degree of practical ability rather than theory alone. Relying solely on traditional classroom education is not enough to produce good programmers. Faced with this challenge and difficulty, novices always ask for help from someone who has more experience or other resources, such as books or the Internet. In turn, it can degrade the motivational state of the student and cause students to drop out from computer science courses (Hoc, 1990; Robins, Rountree, & Rountree, 2003).

Theoretically, knowledge and skill in programming enables the learners to assess their solutions as well as their method of approaching the solutions. This cognitive process facilitates the process of transmitting newly acquired skills and applying them to new problem situations. Realizing this, the development of metacognitive skills for novices has been considered as an essential skill in learning programming. In didactical taxonomies (Mangione, Gaeta, Orciuoli, & Salerno, 2010), metacognitive knowledge refers to the upper level of knowledge that gives the opportunity and ability to the learners to control, comprehend, manipulate and direct their knowledge throughout the learning process. This can be achieved through self-regulated learning individual learning processes (Cleary, Durning, Gruppen, Hemmer, & Artino Jr, 2013; Zimmerman & Schunk, 2001). Possessing such skills, makes the learning process become easier and simplifies the process of knowledge transferal of past problem-solving skills to the problem situation, which consequently increases the potential for the student to learn independently. Theoretically, programming skills enable learners to reflect and formulate solutions. The process of cognition entails modeling, coaching, reflection and articulation (Brown, Collins, & Newman, 1989). Metacognition is a critical skill for students to build and it is a main focus for learning research. It is one of three broad recommendations in "How People Learn" (Bransford, 2000), which lead the general study of learning and education. A number of research works have shown that a strong focus on metacognition in instructional programs can improve the learning outcome of students (Madureira et al., 2014; McGilly, 1996; Palinscar & Brown, 1984). There is potential to produce independent learners by helping them to improve their metacognitive skills and awareness. Many support tools exist to support various skills of self-regulation (Aleven & Koedinger, 2002). However, there are generally few success stories of helping students enhance their metacognitive skills, especially in learning computer programming. This research aims to provide and design activities and scaffold novice programmers by providing an innovative metacognitive supportive learning environment.

1.1 Problem Statement

For many Computer Sciences and IT courses, an elementary pass in mathematics and English is the only entry requirement for a degree program. Introductory computer programming subjects and a few other fundamental subjects may be included in the first semester of study. There are various kinds of bugs in learning to program that either result from or are perpetuated by the minimal use of metacognitive processes during programming. Difficulties have been documented for program writing and reading. These fundamental computing activities can be a challenge to learning and place students in situations to which they are unaccustomed. Knowledge at the conditional or strategic level or higher level knowledge is a knowledge that demands metacognitive skills. Lack of such skills has been stated in several studies in computer science education (Cetin, Sendurur, & Sendurur, 2014; Coull & Duncan, 2011; Linn & Clancy, 1992; McGill & Volet, 1997; Meerbaum-Salant, Armoni, & Ben-Ari, 2013; Volet, 1991). Novices range from being those who never previously experienced programming to those who may have some basic background to programming, attained informally or via formal study in pre-university contexts. A novice programmer in this study refers to someone that lacks programming knowledge. The common mistake for novice learners seems to being prone to miscomprehend the knowledge of programming and approach the programming code line-by-line rather than the conceptual and bigger picture of the knowledge (Soloway & Spohrer, 2013).

The knowledge of the novice tends to be context specific rather than a conceptual view (Kessler & Anderson, 1986; Kranch, 2012; Lee, Rodrigo, d Baker, Sugay, & Coronel, 2011; Soloway & Spohrer, 2013), and they also often fail to adapt and apply correctly the knowledge they have acquired. (McCracken et al., 2001; Scott & Ghinea, 2013; Utting et al., 2013) noticed serious weaknesses in the programming knowledge and skills of the students in introductory courses. As mentioned in the introductory section, classroom teaching and learning is not enough to groom a good programmer. Regardless of what programming, language, method and development environment in the classroom, the concept of programming and logical reasoning processes are challenging for students. In order to solve a variety of problems and contexts of the real world, such knowledge must be applied. As a result, in the context of programming, the rote type of learning is almost impossible. (Berges, 2015; Linn & Clancy, 1992; Soloway & Spohrer, 2013) indicated that these skills seem to be inadequate in newbie learners. (Davidson, Deuser, & Sternberg, 1994) indicated that the metacognitive processes are the key contributors to problem-solving performance in independent domains. (Bransford, 2000) agreed that this skill is a critical process in supporting learning and problem-solving (Allert, 2004). To date, metacognition in programming has been little studied and the above remarks have emphasized the self-regulatory components of metacognition. It is likely that metacognitive activities involving reflection on the state of one's knowledge and skills in programming are central to the development of programming expertise (Mathan & Koedinger, 2005) . For example, careful program documentation should emerge after a programmer discovers the difficulty in tracing program bugs without it. Explicit awareness of the strengths and limitations of one's program design and program debugging strategies should lead one to seek help in order to learn to program more effectively.

The above discussion about the issues and challenges faced by novices are directed at one problem statement, i.e. there is a lack of metacognitive support for novices in learning computer programming. In this research work, the ‘Novice Programmer’ refers to the newbie in Computer Programming. The term ‘support’ means assisting learners in how to use his/her metacognitive skills during learning.

1.1.1 Domain of Computer Programming for Metacognitive Support Tool

Problem-solving is a common activity used as a teaching and learning approach in computer programming because it helps students to develop different cognitive abilities. The emergence of metacognition is a concept that plays a vital role in computer programming problem-solving. It is desirable to choose this domain for the following reasons:

1. Computer Programming is an important subject in Computer Science Education.

    - Computer programming is an essential subject in computer science education and is required to be mastered by those who are interested in studying computer science education.

2. Problem-solving in computer programming involves cognitive skills in a wide range, which creates an environment that is suitable for developing metacognition.
    - The higher level of knowledge, which refers to the knowledge of "when and why", demands the metacognitive skills that are seemingly lacking among the students. This has been reported in several studies (Ismail, Ngah, et al., 2010; Linn & Clancy, 1992; McGill & Volet, 1997; Meerbaum-Salant et al., 2013; Volet, 1991).

1.2 Aims of the Research

This study aims to conduct a series of observations and surveys with the specific research objectives as follows:

1. Identification of the effect of metacognition in learning computer programming
2. Identification of the support features for a metacognitive learning environment
3. Development of support tools based on the identified features in (2) by employing Semantic Web technology
4. Evaluation of the effectiveness and usability of the prototype system

In line with the objectives of this research work, this study aims to find the answers to the following research questions:

R1. How does metacognition affect the learning success of computer programming at university?

R2. What is the role of metacognition in teaching and learning of computer programming?

R3. What are the characteristics of metacognitive instruction that should be incorporated in the support system in learning programming?

R4. How can a metacognitive support environment benefit novice programmers in learning computer programming?

1.3 Related Work

Based on the reviewed literature, in general, there is a lack of research work on a metacognitive support system for novice programmers in learning computer programming. In addition, no similar work using the Semantic Web as the underlying technology for building this support tool could be located. The use of Semantic Web technology in education as a method of content specifying has been explored by recent studies in education (Ghaleb et al., 2006; Kramarski & Mevarech, 2003; Sampson, Lytras, Wagner, & Diaz, 2004; Stojanovic, Stojanovic, & Volz, 2002) to enhance the mechanism for the retrieval of learning resources. The work by (Ghaleb et al., 2006), for example, employed the technology of the Semantic Web in the E-learning model by providing services and a tool in the context of a semantic E-learning portal. The work by (Mangione et al., 2010) provides an educational environment by extending the E-learning platform with the Semantic Web and social web technologies and methods. Other studies (Stojanovic, Staab, & Studer, 2001) present an approach for developing E-learning using the technology of the Semantic Web that is mainly based on the content of ontology-based descriptions, structure and the materials learning context and thus provides flexible access and is personalized to these learning materials. Over the years, there has been much work and discussion on providing an environment, especially in E-learning, to support novice learners. The likelihood of enhancing the retrieval mechanism (e.g. produce search based on meaning) is the benefit of using the Semantic Web. Thus, utilizing Semantic Web technology in a metacognitive based E-learning system would contribute to an interesting and novel research, especially in supporting teaching and learning programming metacognitively.

1.4 Research Methodology

The educational psychology theory adopted in this research work and the quantitative methods are used to solicit and elicit data. This research work involves the following activities:

i. Conducting a literature study relating to metacognition in the field of educational psychology that is associated with a supporting environment for novice programmers in learning programming.
ii. Conducting interviews with expert lecturers to explore the awareness of metacognition and the implementation of metacognitive strategies in teaching introductory computer programming courses at Universiti Teknologi MARA (UiTM), Shah Alam.
iii. Conducting an empirical study using a well-tested and validated instrument called Metacognitive Awareness Inventory (MAI) to investigate the relationship between metacognition with learning success in learning computer programming at the University.
iv. The Metacognitive Support System for Novice Programmer (MSSNP) development using Semantic Web technology.
v. The MSSNP System Prototype assessment and evaluation.

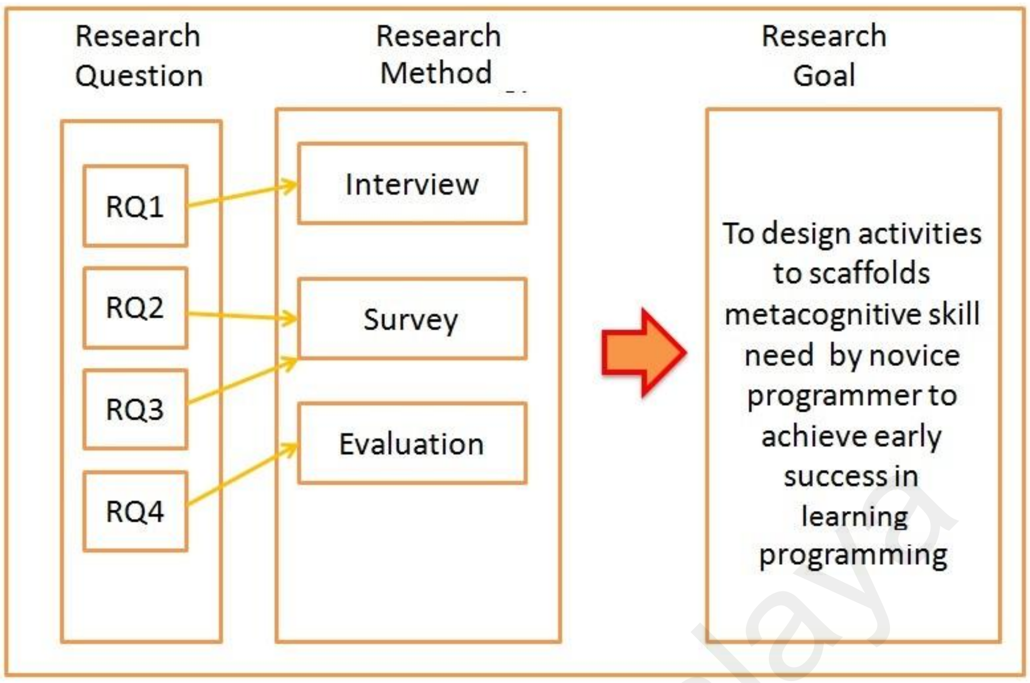

Research Question
Research Method
Research Goal
RQ1
RQ2
RQ3
RQ4
Interview
Survey
Evaluation
To design activities to scaffolds metacognitive skill need by novice programmer to achieve early success in learning programming

Figure 1.1: Research Questions, Methods and Goal

1.5 Research Scope

The research scope study concentrates on developing a support system for novice programmers by putting the metacognitive activity elements in learning programming using the Semantic Web technology. Thus, the scope and boundary of this research work are as follows:

1. Novice programmers in this study refer to students who have already taken the Fundamentals of Computer Problem-solving subject specifically in the Faculty of Computer & Mathematical Sciences (FCMS), University Teknologi MARA.
2. Metacognitive awareness Inventory (MAI) is used to investigate the relationship between metacognition with the learning success in learning Computer Programming at University
3. The development of a prototype system that is perceived to be beneficial for novice programmers to improve their metacognitive skills in learning Computer Programming.

CHAPTER 2: METACOGNITION IN LEARNING PROGRAMMING

The major philosophical and theoretical underpinnings of this research are presented in this Chapter. It begins with the introduction of metacognition, and the examination of definitions, which focuses on different aspects of this interesting phenomenon. This chapter also presents a model that expands the insight of what metacognition is and describes the chosen model and emphasizes knowledge monitoring, which is one of the metacognition skills. The measurement methods of metacognition are discussed together with the assessment tools that are employed in this research work. It then re-examines existing studies related to the problems in computer programming education, the relation of problem-solving in computer programming with metacognition, the characteristics of novice programmers and the difficulties faced by them, as well as presenting a review of the research that interplays between metacognition, problem-solving and computer programming. The approaches to metacognitive instruction are also described in this chapter (the necessary metacognitive elements that are possible to embed in the instructional system). This chapter ends with the proposed underlying technology used to build the prototype system.

2.1 Metacognition Definition

"Metacognition" is a term used in the area of educational psychology. It comes from the root word ‘meta’ – to define ‘thinking about thinking’ or ‘knowledge about knowledge’. Without realizing it, we apply metacognition in our daily activities every day (Briñol & DeMarree, 2012; Reder, 2014). The active control of cognitive processes that we engage in during learning can be referred to as metacognition; which involves activities like planning, monitoring and evaluating knowledge. To be a successful learner, metacognition is the key, as it is related to intelligence (Borkowski, Carr, &

Pressley, 1987; Carr, Kurtz, Schneider, Turner, & Borkowski, 1989; Frith, 2012; Pishghadam & Khajavy, 2013; Song, 2013; Thompson, Prowse Turner, & Pennycook, 2011). Therefore, it is important to research the elements of metacognitive activities and their development so that a learner can better apply their cognitive resources appropriately.

"Thinking about thinking" is the definition most often used to describe metacognition. In actuality, it is not that simple to define metacognition. Although the term has existed in the vocabulary of educational psychologists for the past few decades, there is still much polemic over what exactly metacognition is. There are several terms often used interchangeably in the literature to describe this interesting phenomenon (e.g. meta-memory, self-regulation, executive control), which is one of the reasons behind the confusion. Although there are some differences between definitions (see (Haynie, Shepherd, & Patzelt, 2012; Van Zile-Tamsen, 1994; Van Zile-Tamsen, 1996)), all emphasize the executive role processes in the monitoring and regulation of cognitive processes.

Flavell has always been linked to the term "metacognition", and, according to him, it is composed of both knowledge and regulation of metacognition (Flavell, 1987; Flavell, 1979). Acquiring knowledge about the cognitive process is a simple way to describe metacognitive knowledge, which can be employed to control the cognitive processes. Flavell categorizes three types of metacognitive knowledge – variables that relate to the knowledge of the person, variables that are associated with the task, and strategy.

2.2 Differences between Metacognitive and Cognitive

In Flavell’s models (Flavell, 1979; Lee, Lim, & Grabowski, 2010; Winne, 2011), he assumes that metacognitive and cognitive are distinct in their function and content, but that there is a resemblance in their quality and form, as much as they can be possessed, corrected, forgotten as well as can be thought of in an external method. This is similar to cognition in that the information can be correct or incorrect, shared, validated or subjective. Hence, cognition and metacognition can be differentiated using two basic features, content and function. For instance, for people who have difficulty in understanding the principles of mathematics, would this be considered as metacognitive or cognitive knowledge? (Flavell, 1979; Lee et al., 2010) acknowledge that the distinction between these two types of knowledge lies in how it is used.

Metacognition involves the process of inspecting whether the cognitive goal could be achieved or not (e.g. understanding a text), whereas to ensure that the objective and goal accomplished have been reached is the role of metacognitive strategies (e.g. self-questioning to understand text). Cognitive activity is usually preceded or followed by metacognitive experiences. In many cases, they happen when cognition activity fails, for example the failure of what one just read. To activate the metacognitive processes, one needs to attempt to rectify the situation (Dorça, Lima, Fernandes, & Lopes, 2013a, 2013b; Roberts & Erdos, 1993).

Depending on the metacognitive or cognitive purpose, questioning, for example, could be regarded as either a metacognitive or a cognitive strategy and these two strategies may overlap. For example, for obtaining knowledge, the self-questioning strategy is used, and can be referred to as a cognitive process, while for monitoring what has been read can be considered to be a metacognitive process. Any attempt to inspect one without recognition of the other would provide an inadequate picture because these two strategies (cognitive and metacognitive) are dependent upon each other and closely related. In other words, metacognitive can also be considered as knowledge that is actively used in ensuring that the goal is met. For instance, a novice programmer may use planning in how to approach a programming problem, "I know that I have difficulty (person variable) with programming logic (task variable), so I will do the pseudocode first before coding (strategy variable). Information of knowledge (strength and weakness as well as the nature of the task) without its application is not considered as metacognitive".

The metacognitive perspective results from learning experiences and is able to activate responsive processes on knowledge as well as self-knowledge (Nett, Goetz, Hall, & Frenzel, 2012; Tsai, 2009). Metacognitive strategies is presented as the highest level in didactical taxonomies, which refers to the learners' ability to understand, manipulate and direct their learning knowledge. Self-regulated learning can be used to achieve metacognition and is considered as the learner's self-directed management of learning (Van den Boom, Paas, van Merriėnboer, & Van Gog, 2004).

Until today, there are relatively few stories that successfully help students to improve their metacognitive skills, particularly in learning programming using an interactive and reflective learning environment. One of the instructional design principles proposed by (Bransford, Brown, & Cocking, 2000) is "how people learn" is to scaffold metacognition skills and awareness. Having better metacognitive skills encourages them to be self-regulated learners, and, furthermore, can provide them a better learning environment across domains and contexts. Evidence shows that well-designed instruction of metacognition has a positive impact on metacognitive behavior, and, consequently, on domain learning, for example, instruction on the use of debugging skills (Askell-Williams, Lawson, & Skrzypiec, 2012; Carver & Mayer, 1988; Mayer, 2011) or self-regulatory strategies (Bielaczyc, Pirolli, & Brown, 1995).

2.3 Metacognition Models

Metacognition conceptualization has attracted a broad range of researchers who have proposed and designed a number of models of metacognition. Some are more conceptual and have resulted in the development of a theoretical framework, such as Brown's and Flavell's model of metacognition; while other researchers focused on particular aspects of metacognition, such as metamemory and memory processes (Mazzoni & Nelson, 2014; Ornstein, Baker-Ward, & Naus, 1988; Schneider, 1985), knowledge of the language structure (Tunmer & Bowey, 1984), studying and learning from texts (Brown A., 1987; Brown, 1980) and self-regulation metacognitive strategies during reading (Scardamalia & Bereiter, 1985).

The foundation of theory for metacognition is presented by Flavell's model (defined metacognition components and their interaction). Similarly, with Brown's model, which emphasized the distinction of these two different categories of metacognition (knowledge of cognition and regulation of cognition). Finally, Tobias and Everson's Model is presented in the following section, as they propose a standardized module of metacognition for flexible arrangement, which is used as the underlying theory for this study.

2.3.1 Tobias and Everson's Model

Tobias and Everson indicated that metacognition is composed of knowledge of cognition, learning processes, cognitive monitoring and the control of these processes. These components are organized into hierarchical order in which the knowledge monitoring skill is a primary process to stimulate the other metacognitive skills, as presented in Figure 2.1.

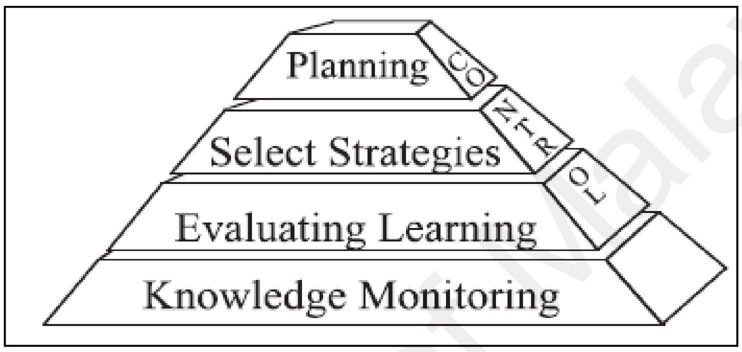


Figure 2.1: Tobias and Everson – metacognition model hierarchy (Tobias, Everson, Laitusis, & Fields, 1999)

They identified knowledge monitoring as knowledge of awareness of what one knows and what one does not know. Thus, those who are able to differentiate accurately between what is yet to be attained and what they have already learned benefit as they can use time wisely and concentrate most of their energy on topics that are unfamiliar to them. Conversely, they argue that those who lack the skill of knowledge monitoring are prone to allocate their time as well as their resources inefficiently, which, consequently, contributes to their greater difficulty in mastering new knowledge (Tobias, Everson, & Laitusis, 1999).

The monitoring aspect of metacognition has been investigated extensively by Tobias and Everson who made the assumption that accuracy in monitoring knowledge is important in the context of training and learning where students are required to master new knowledge (Tobias S. et al., 1999). A series of empirical studies was conducted by them to investigate the relationship of learning from instruction to the monitoring aspect of metacognition in various domains and measurement concerns. They emphasized such issues as knowledge monitoring on domain specificity and the relationship of academic ability to knowledge monitoring. The metacognitive awareness instrument developed by Tobias is detailed in section 2.4.2. This model is generally appropriate for this research work because it allows emphasis to be given to the specific metacognitive skills that are relevant to problem-solving.

2.3.2 Flavell's Model

There are four components defined by Flavell's model (Flavell, 1979) in metacognition: (1) knowledge of metacognitive, (2) experience of metacognitive, (3) goals or tasks, (4) strategies or actions, and the effect of the interaction between these components gives the ability to the person to control and monitor a wide variety of enterprises of cognition. The components relationship is illustrated in Figure 2.2.

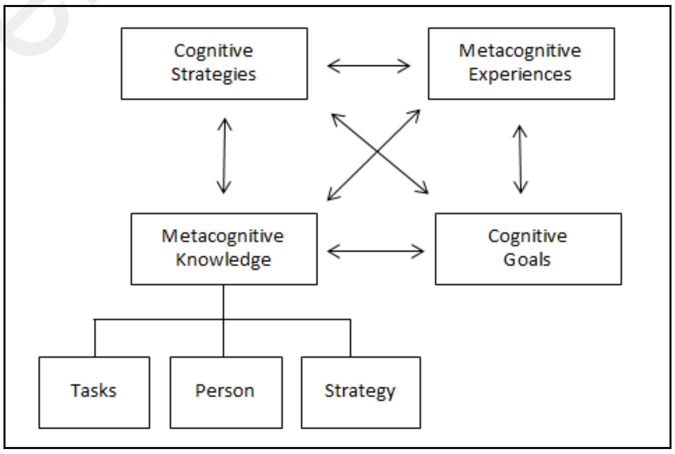


Figure 2.2: Flavell's model of metacognition (Flavell, 1979).

The definition of Flavell's model components can be summarized as below:

Metacognitive knowledge can be defined as extensive knowledge about the processes of cognition that one acquires for cognitive abilities. Flavell indicates that knowledge metacognition affects the course as well as the outcome of cognitive enterprise, which results from the interaction of various factors or variables (Flavell, 1979). (Flavell, 1979) identifies three categories of factors according to person, task and strategy, as described below:

- Person – The person category is composed of a cognitive processor, which relates to the beliefs and knowledge that one has regarding the differences between the differences of intra-individual (e.g. the fact that one is better at memorizing text than calculating), or the recognition of universal properties (e.g. belief or knowledge that there is a dissimilarity in the kinds and degrees of understanding).
- Task – Information about a specific cognitive task in which a person is engaging. In this category, one would get insights into the implications of the information being presented and whether it is being well presented or poorly presented, as well as the goal setting.
- Strategy – Process or knowledge of finding strategies that are likely to be effective in achieving the goal in various cognitive tasks
- Metacognitive experience – Efficient cognitive action coming from effective experience. It is the success or the failure in learning or other cognitive enterprise, such as feeling confused about a text passage

2.3.3 Brown's Model

In Brown's model (Baker & Brown, 1984), metacognition can be categorized into two main components: 1) knowledge of cognition involves conscious reflection of one's cognitive abilities, and 2) regulation of cognition concerns the self-regulatory mechanisms during the ongoing process of problem-solving. These two divisions of metacognition are closely related and recursively complement each other (See Figure 2.3).

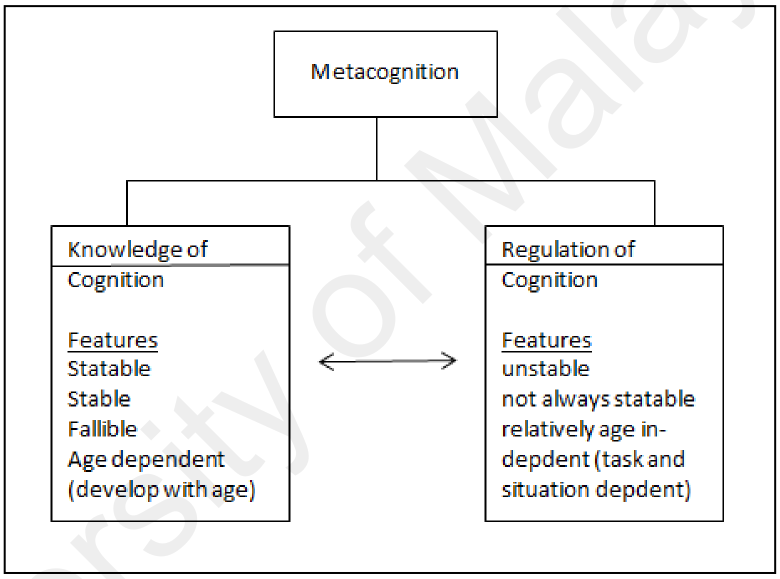


Figure 2.3: Brown's Model of Metacognition (Baker & Brown, 1984)

Regulate learning as well as oversee learning are the two functions in terms of the regulation of cognition in which the activities include planning (e.g. scheduling strategies, predicting outcome, experimenting and prior to problem undertaking, inspecting (e.g. testing, monitoring, revising) and checking outcomes (evaluating the outcome against the criteria). Knowledge of cognition can be stable or late developing, which relates to the cognitive processes of the individual. On the other hand, regulation of cognition can be relatively rarely stable, unstable and independent of age. Self-regulatory behavior may be different from one person to another, from one situation to another situation because regulation may be affected by self-concept (e.g. self-efficacy, self-esteem) and pattern of arousal (interest, anxiety, fear).

2.4 Metacognition Assessment

(Garner & Alexander, 1989) focus on the relevance for measuring metacognition with empirical research, suggesting how the question “How can knowledge about knowledge be measured accurately?” should be addressed. Many researchers (e.g. (Semerari et al., 2012; Veenman, 2011; Wilson & Bai, 2010)) have come up with ideas and designed methods and instruments to measure all the components of metacognition and were tested across domains. This approach to measuring metacognition ranges from self-report questionnaires (one rates one’s own knowledge of metacognition) to verbal-report (one recalls what one’s thoughts are during the learning experience). The reliability and the accuracy of these methods in measuring metacognition has attracted the attention of researchers and they claimed that all such methods are fallible because of the complexity in measuring metacognition. Hence, to provide a more reliable measurement for the investigation of phenomena, multiple methods have been suggested by many people to avoid a common source of error (Greene, Robertson, & Costa, 2011; Schellings, van Hout-Wolters, Veenman, & Meijer, 2013; Veenman, 2011). For example, verbal reports can be combined with performance measurements as well as combining verbal data with nonverbal data. In this research work, we have employed two types of method for measuring metacognition, that is, using Metacognition Awareness Inventories (MAI) and Knowledge Monitoring Assessment (KMA), both of which will be discussed in the following section.

2.4.1 Metacognition Awareness Inventories (MAI)

Since it was first utilized in the 1970s, the concept of metacognition has become quite fashionable and several efforts have been made by researchers to design a metacognitive inventory. To determine the level of metacognitive awareness in learning programming among novice programmers, an Inventory developed by (Schraw & Sperling Dennison, 1994) is used. This 52-item inventory has a comprehensive scale assessing various facets of metacognition, including metacognitive knowledge and regulation (Schraw & Sperling Dennison, 1994). Items are classified into eight components under two primary categories, (1) knowledge of cognition and (2) regulation of cognition(see Figure 2.4). There are three types of knowledge in the knowledge of cognition: declarative knowledge, procedural knowledge, and conditional knowledge. Hence, the regulation of cognition can be defined as a set of processes and activities that aids students to control their learning that consists of five types of knowledge: planning, debugging strategies, information management strategies, evaluation and comprehension monitoring. A strong connection between these factors suggests that both types of knowledge (knowledge of cognition and regulation of cognition) may work simultaneously to help students become self-regulated learners. The MAI is used to answer the hypothesis put forward in this research work – 'Metacognitive skills influence students learning success in Introductory Computer Programming at University'. This is the main hypothesis driving this research work in providing an instructional tool to scaffold the metacognitive skill of Novice Programmers in learning Computer Programming'.

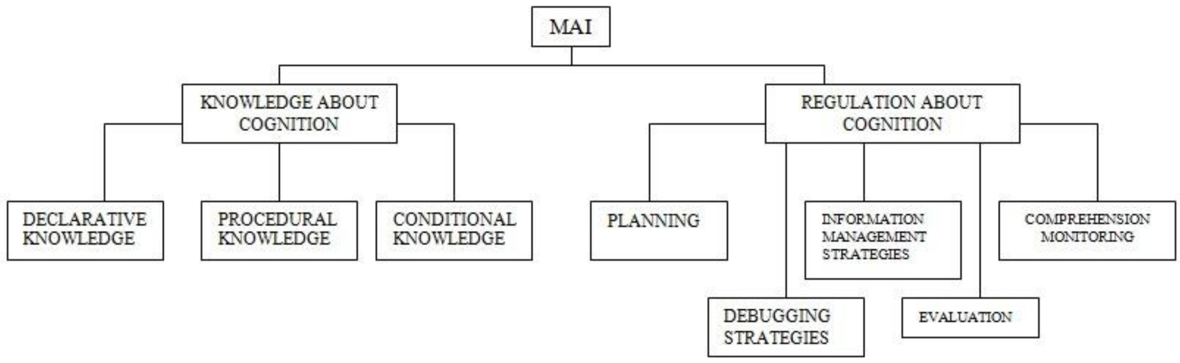


Figure 2.4: The decomposition of MAI (Schraw & Sperling Dennison, 1994)

2.4.2 Knowledge Monitoring Assessment (KMA)

(Tobias & Everson, 1995) developed an instrument of assessment that focuses on knowledge monitoring, part of the metacognition component that could be applied across domains. In terms of its applicability, the instrument has indeed been shown to be quite general and has been used successfully across multi-disciplines (Cella, Swan, Medin, Reeder, & Wykes, 2014; Goh & Hu, 2014; Wu, Valcke, & Van Keer, 2012). The techniques concerning how the knowledge monitoring assessment works is by first asking the subjects whether they have knowledge about something or not and later challenging them to prove their assessment that can be right or wrong. For instance, a student is asked about his/her comprehension of a given word and later asked to provide the definition. In the scenario of task oriented, the programming problem presented to his/her estimation of her knowledge concerning whether or not she can solve that problem, and, at some later point in time, asking her to solve the problem. After answering all the questions, the student is submitted with a preview for each question. The knowledge monitoring assessment provides statistical profile of the individual by gathering a significant number of elementary assessments of that person. The four scores generated by KMA are as follows:

1. Student estimated her ability of the knowledge and performed accordingly ((a) abbreviated as ++)
2. Student estimated her inability to solve problem given, but when challenged, she succeeded ((b) abbreviated as -+)
3. Student estimated her ability of the knowledge but when challenged, she failed to perform ((c) abbreviated as +-)
4. Student estimated her inability to solve problem given and failed to perform ((d) abbreviated as --)

Each time these scenarios happen is counted for the same student with the four cases being labelled as (a), (b), (c) and (d), as presented in Table 2.1. The simple formula that represents the Knowledge Monitoring Assessment (KMA) score is then obtained as follows:

$$((a+d)-(b+c))/(a+b+c+d)$$

Table 2.1: KMA possible value. Types of Outcome and their respective counts

Actual Performance	**Student Assessment**	
	Know	**Do not know**
succeeded	(a)	(b)
failed	(c)	(d)

2.4.3 Methods of Metacognition Assessments

Recently, many diagnostic tools have been designed to assess and measure metacognition. The trend of those tools is addressed to assess the prospective as well as retrospective of metacognition to specific arithmetical performance. Self-reporting, hypothetical interviews and questionnaires are some examples of the prospective methods, in which one has to indicate to what extent a statement on a Likert-type of scale (e.g. ‘I do self-questioning to know the meaning of the text I’ve read’) is the representation of behavior (e.g. (Elshout-Mohr, Meijer, van Daalen-Kapteijns, & Meeus, 2003)). In retrospective techniques, for assessing metacognition, both interviews and questionnaires have been applied. However, there is a risk of distortion of memory with the retrospective type of assessment due to the time lag between the problem-solving and actual performance as well as the verbal reports afterwards. Think aloud protocols that are also categorized as concurrent assessment can take place as an addition to both techniques. Throughout the implementation of these protocols, instruction is given to the participant to merely verbalize their thoughts during task performance. Table 2.2 provides the techniques in assessing metacognitive skill and awareness.

Table 2.2: Metacognitive Assessment Comparison

Technique	**Description**	**Advantage**	**Limitation**
Concurrent think-aloud (Greene et al., 2011; Rosenzweig, Krawec, & Montague, 2011; Schellings et al., 2013)	Learner expresses out loud everything crossing his/her mind and everything that occurs while performing a task	Rich data can be extracted from the process that is invisible to other techniques	Automated processes remain inaccessible; Verbalizations might be a problem to some targeted subjects; Extensive analysis of data is needed
Post-performance interviews (Dabarera, Renandya, & Zhang, 2014; Ford & Yore, 2012)	Interview learner past learning experience	Provides data from responses to specific, direct probes.	Awareness deficiencies in processing that are automated; cognitive events that happened during the time of processing and reporting but always failed to remember; lack of verbal fluency
Cross-age tutoring (De Backer, Van Keer, & Valcke, 2012; Hill & Greive, 2011)	Conduct tutoring session to observe which strategies and behaviors will encourage solving a problem – younger children	Non-verbal data; avoid subject speculating what the investigator desires to hear and answering accordingly.	Suitable for certain type of strategies (e.g. awareness of usefulness of text re-examination strategy).

Self-Report Inventory (Baas, Castelijns, Vermeulen, Martens, & Segers, 2014; Veenman, 2011)	Self-questionnaire typically using formats like Likert scale, forced choice or true-false	Structured and convenient: ease of use and secure	Answers given may just be pleasant to the investigator; Answers about partially automated processes are difficult to acquire

2.5 Problems in Computer Programming Education

Difficulty in computer programming is common for Malaysian students. Studies have shown that analytical skills and problem-solving are the two major problems regarding computer programming (Henderson, 1987; Linn & Clancy, 1992; Poli & Koza, 2014; Soloway & Spohrer, 2013). However, according to (Ismail, Azilah, Naufal, & Kelantan, 2010), many students are "woefully inadequate" in terms of problem-solving skills. (Soloway & Spohrer, 2013) agreed that analytical thinking and problem-solving skills are major weaknesses of students in a computer science course, and a major theme of a computer science course should be emphasized in these skills. Problem-solving in computer programming study involves cognitive skills that provide practice for students to work in a methodical manner as well as to build representations (Mayer, 2013), which creates an environment that is suitable for developing metacognition. Programming foster students to evaluate their thinking process as well as their solutions is a cognitive activity that facilitates the process of applying a newly acquired problem to novel problem situations. Regardless of the approach adopted in problem-solving, it is recognized as being an essential part and considered as the first step taken in programming. In addition, errors correction in programming also contributes to knowledge building (Sternberg & Frensch, 2014). The competency of programmers can be developed through thoroughly organized programming knowledge and good programming skills (Linn & Clancy, 1992). The prior practices and knowledge are also attributed to the difficulty in programming. In addition, the difficulty of programming not only cause a difficulty in understanding the concepts or syntax, but also planning (Costa, Aparicio, & Cordeiro, 2012; Havenga et al., 2012). Students may be good at giving an explanation and their understanding of the concept of programming (e.g. the function of pointer), but still be unable to apply it appropriately. Most computer programming introduction courses only emphasize declarative knowledge (lower level knowledge) and procedural knowledge (know what and how") (Su, Yang, Hwang, Huang, & Tern, 2014), as a result, they are unable to explain and often fail to understand the sematic actions. Traditional approaches of learning, such as reading, lecturing and practical (Hazzan, Lapidot, & Ragonis, 2011) are the most common approaches to learning computer programming. These types of learning environment only produce passive students, especially in larger groups, because of the minimal interaction between students and teachers. (Uysal, 2014) Suggested that to have effective teaching, computer programming must be used together with problem-solving strategy instruction. (Mason, 2012) also suggested that lessons of computer programming must employ systematical instruction design activities that are rich in practice opportunities and feedback.

2.6 Metacognition and Problem-Solving

(Frensch & Funke, 2014) related the capacity to solve problems as the main aspect of intelligence. Without having to know in advance how to solve the problem, one usually always solves problems to achieve something. Factors that make up the problem-solving are different from person to person (Bardach, 2011; McNie, 2007). What is demanding and challenging to one person might be easy and simple to another. The demands of performing the task and the interaction of one's experience are the activities involved in problem-solving. Research has shown (Jacobse & Harskamp, 2012; Sandi-Urena, Cooper, & Stevens, 2012; Scherer & Tiemann, 2012) that those who consciously utilize their intellectual skills as well as showing perseverance and flexibility in problem-solving can be considered to show well-developed metacognitive abilities. It has also been pointed out by researchers (Bulu & Pedersen, 2012; Sandi-Urena et al., 2012) that metacognitive regulatory skills are important skills in problem-solving (e.g. planning and monitoring). In this research work, we have chosen problem-solving as the activity of learning upon which to build the proposed system. A such, we emphasize those metacognition aspects that are most employed in problem-solving and the monitoring of knowledge, which includes comprehension of the problem, planning (strategies and heuristics) and evaluating the process of problem-solving.

2.6.1 The Characteristics of Problems and How Metacognition Helps

There are three important attributes in problems: givens, obstacles and a goal. The givens can be defined as the elements, the conditions and their relations that create the problem-situation's initial state. The obstacles refer to the characteristics of the problem situation and problem solver who finds difficulty in transforming the problem initial state into the desired state. Whereby the goal can be referred to as the desired outcome or solution of the problem. Transforming the problem initial state into the desired one is an active process (Davidson et al., 1994; Eysenck, Ellis, Hunt, & Johnson-Laird, 1994). In general, with the aid of metacognition, the problem solver would be able to:

1. Acknowledge the problem to be solved
2. Grasp and understand the problem
3. Grasp the idea of how to reach the solution.

The metacognitive processes help individuals pinpoint the givens, goals and obstacles of problem-solving. (Davidson et al., 1994; Eysenck et al., 1994) guide the active process of converting the problem initial state into the desired state, and, with this in mind, they require the identification of the problem as well as the representation of the problem, planning how to move on, and evaluation of the obtained solution (Schraw & Gutierrez, 2015). Preparation to solve the problem, understand the problem as well as the desired goal and givens is the crucial part in solving the problem because without adequate understanding, the correct solution cannot be generated. This process is called the familiarization stage (Halpern, 2014). The actual problem-solving stage is called the production stage (Halpern, 2014). During this stage, the solution paths are produced by the problem solver to define the problem space. The evaluation stage involves the activity of verification of the problem-solving (Halpern, 2014). During this stage, the path solution is evaluated by the problem solver in order to choose the best one (Halpern, 2014).

2.6.2 The Interplay between Problem-Solving, Metacognition and Computer Programming

According to Sternberg and Sternberg (Sternberg & Frensch, 2014), general problem-solving steps include: problem identification, problem definition, strategy formulation, organization of information, allocation of resources, monitoring and evaluation. The solving of programming problems requires similar steps. It comprises the implementation of abstract ideas, plans and/or designs by means of expressions, statements and programming constructs (syntax) in such a way as to resemble the correct logic and meaning of those expressions, statements and constructs (semantics), and to solve the programming problem effectively. The role of metacognition in the solving of programming problems is imperative. (Bergin, Reilly, & Traynor, 2005) found that students that possess metacognitive management skills and strategies perform well in programming compared to lower-performing students. In fact, the more complex a programming problem is, the greater the need for metacognitive control, purposeful reflection and positive feedback (Havenga, 2011). A programmer needs to apply in-depth reading skills and meta-comprehension to judge how clearly and effectively he or she understands the programming problem.

Furthermore, programmers need to direct their problem-solving processes, apply a programming approach, correct programming errors, think deeply about their programming solutions and test program output. These problem-solving steps require metacognitive control, such as planning (planning the solution), monitoring (monitor the design and development of the program) and evaluation (test and reflect on the programming solution). Students should therefore manage their programming processes, motivate their decisions, articulate their actions and investigate alternative solutions to improve the quality of their programs. The teacher has thus the responsibility to support students in developing metacognitive skills and applying these during problem-solving and program development.

2.7 Novice Programmer

Novice programmers can be defined as a person that lack the knowledge and programming expert skills. Several factors and elements have been described in the literature and also reviewed by (Costa et al., 2012; Havenga et al., 2012; Rist, 1996). Limitations in seeing the abstract knowledge of the program is the common area that novices face as they approach programs by line-by-line of written code rather than the big picture of program structures. A common characteristic that novices seem to experience is that they have limited ability to see knowledge in abstract; the knowledge of novices tends to be context specific rather that general (Kessler & Anderson, 1986), and they also fail to correctly apply the knowledge they have obtained. In fact, an average student does not usually make much progress in an introductory programming course (Kölling & Rosenberg, 1996). This was also realized by the study of McCraken et al. (Mayer, Dyck, & Vilberg, 1986), who noticed serious shortcomings in student's programming skills in fundamental courses. There are effective and ineffective novices, i.e. students who learn without excessive effort and those who do not learn without inordinate personal attention. In programming courses, different student behaviors in confronting a problematic situation can be recognized. Perkins (Papert & Solomon, 1971) named two main types: stoppers and movers. In a problematic situation, stoppers simply stop and abandon all hope of solving the problem on their own, while movers keep trying, modifying their code and using feedback about errors effectively. Naturally, student's personal learning strategies and motivation affect their success in learning programming strategies.

2.7.1 The difficulties encountered by Novice Programmers

Learning programming is generally considered hard, and programming courses often have high dropout rates. It has even been said that it takes about 10 years for a novice to become an expert programmer (Soloway & Spohrer, 1989; Soloway & Spohrer, 2013). Educational research has been carried out to determine the characteristics of novice programmers and to study the learning process and its connection to the different aspects of programming. The process of the progress of the learner from novice to an expert can be viewed as a process of learning. The dissimilarity of novices and experts have been investigated by researchers and yielded significant information about how to guide novices programming knowledge. Firstly, novices have grammatical problems in programming language, and, hence, they are battling with syntactic knowledge (Lahtinen, Ala-Mutka, & Järvinen, 2005). Experts, in the context of semantic knowledge possess an effective notional or virtual mental model compared to novices who have yet to build such models. In terms of schematic knowledge, experts use deep structure to program categorization based on required routines. In contrast, the approach used by novices for categorization is that of superficial features as they are unskilled at problem decomposition and tend to use low-level plans. In contrast, experts decompose problems into manageable sub-problems where they keep the overall view of the problem in mind as well as consider many alternative solutions compared to novices (Hu, Winikoff, & Cranefield, 2012; Lahtinen et al., 2005). Hence, to counter the challenges in computer programming, especially to novice programmers, providing good strategies for instruction are important for optimal learner support (Soloway & Spohrer, 2013).

2.7.2 The essential need of Metacognition skill in Novice Programmers

As indicated in the literature, teaching programming is a universal problem in computer science courses (Soloway & Spohrer, 2013). The central development competency of a programmer is the problem-solving skills (Soloway & Spohrer, 2013), and yet, these skills seem inadequate. (Soloway & Spohrer, 2013) also note that analytical thinking and problem-solving are major weaknesses in programming. Theoretically, the evaluation of thinking as well as the solution are the cognitive processes that are encouraged in programming that facilitate the process of transferring the new skills of problem-solving to the novel situation of the problem, however, it is difficult to achieve the outcome. The process of metacognition is an important contributor to the performance of problem-solving across a wide range of domains (Azevedo & Aleven, 2013; Davidson et al., 1994). According to (Bransford et al., 2000; Brown, 1980), novice programmers have metacognitive deficiencies regardless of their age. In addition, they fail to reflect on the design approach solution (Berliner & Calfee, 1996). Thus, to be a lifelong learner and become a successful problem solver, it is necessary to develop articulation as well as reflection in novice programmers.

2.7.3 Emergent requirements of a Support Environment for Novice Programmers

It is proven from the review of existing tools that support tools may be used to help novice programmers in learning. This has prompted researchers to develop tools and methods to mitigate the difficulty of learning computer programming. There are a broad range of methods and approaches to teach the complexity of computer programming that interleave with the concepts. Hence, a support tool that is adequately flexible to provide many different forms of instruction by incorporating metacognitive learning elements are of the most value. However, to date, very few attempts have been made to provide an adequate learning tool for novices to improve their metacognitive skills. Table 2.3 presents a summary of research that provides a support learning tool for novice programmers in learning programming.

Table 2.3: Research on support learning tools for novice programmers

Research	Description
A program design tool to help Novices learn programming (Garner, 2007)	The development, use and evaluation of a tool that helps novice programmers generate pseudocode designs for simple programming problems. It discusses the difficulties of learning to program; what is meant by pseudocode; the development of the tool; and the evaluation of the tool with students.
Personifying Programming Tool Feedback Improves Novice Programmers Learning (Lee & Ko, 2011)	Presented a Gidget, a game where the eponymous robot protagonist is cast as a fallible character that blames itself for not being able to correctly write code to complete its mission. Players (Novice programmers) learn programming by working with Gidget to debug its problematic code.
SNOOPIE : Development of a learning support tool for novice programmers within the conceptual framework (Coull, 2008)	Framework for learning the Java programming for novice programmers.
Use of CALMS to enrich learning in an Introductory Programming Course (Thota & Whitfield, 2009)	Computing Augmented Learning Management System. Integration of computer science related content with the University specified learning management system – learning theories, instructional process and learning taxonomies – educational media and the organization of the learning content.

A novice programmer's support environment (Liffick Blaise W & Aiken Robert, 1996)	Presents a model of how programming knowledge can be represented by five cognitive levels: lexical, syntactic, semantic, schematic, and conceptual.

2.8 The Instructional Metacognitive Activities Design

The most effective approaches of metacognitive instruction involve theory and practice. The knowledge cognitive processes, learning strategies and opportunities to apply both metacognitive and cognitive strategies must be given to a learner as well as the outcome of their efforts for self-regulation development (Boyle, Rosen, & Forchelli, 2014; Brown A., 1987; Garrison & Akyol, 2013). Knowledge alone without experience or vice-versa is not sufficient for metacognitive control development. Hence, it is necessary for the instructional context to consider a design for embedding metacognitive activities. (Bannert & Reimann, 2012; Lin, 2001; Mishra & Director, 2013) suggest that both social aspects and cognitive student development should be the focus when designing metacognitive activities that relate to knowledge about the self as the learner as well as about specific domains. (Lin, 2001) affirms that this approach is balance theoretically and practically Strategies to promote metacognition in the classroom have been identified by researchers. An interesting set of tasks and activities that can be adapted in different situations was proposed by (Angelo & Cross, 1993). For more effective development of learning strategies, it is important to assess students' abilities in metacognitive and target instruction. Representational tools and graphics for learning are the typical support of metacognitive development provided in a computer based learning environment that encourage the help-seeking behavior and self-reflection (Aleven, Roll, & Koedinger, 2012; Garrison, 2011; Laurillard, 2013; Reusser, 1993). Guided questioning and reflective prompts are some of the examples of self-reflective activities that require learners to express their ideas and knowledge, as well as explicit self-explained prompts, and self-assessment support features (Gan & Hattie, 2014; Ifenthaler, 2012; Katz, 2002; Latva-karjanmaa, 2001; Molenaar, Roda, van Boxtel, & Sleegers, 2012). Some of the instructional learning environments may apply many approaches to deliver metacognitive strategies in learning. (Murray et al., 2013; Puntambekar & Du Boulay, 1997), for example, applied peer collaboration, reflective prompts and self-questioning.

2.8.1 Reflective Questions and Prompts

Reflective questions and prompts are simple ways used by educators and researchers for interactive discussion throughout the teaching and learning process. The revision of the learning experience toward critical thinking as well as an action plan is always the initial part of reflective activity (Nickerson, 2012). This activity encourages the self-reflection of learners, which can be used to perform tasks in learning (e.g. solving a problem) as well as to justify the reasons for the strategies employed. There is a dissimilarity between prompts and questions. In nature, questions are more general, serving as an approach for triggering broad metacognitive control and monitoring (e.g. So what or Now what?). Whereas prompts are more specific as well as more directive that provide assistance for learner on specific aspects of the learning processes. This may remind students about what task has to been done and what the next steps are, as well as make a relationship between past experience and the existing task.

Prompting is a method used in instruction for supporting and guiding learner regulation in the problem-solving process (Bannert & Reimann, 2012). As suggested by previous researchers (Davis, 2000; Lai, 2008), question prompts can be an effective approach to inculcate reflection as the result of the cognitive complexity in which the learner feels about, thinks about and makes relations in experience. Learners could develop their understanding and locate the important tasks in a larger context by engaging in reflective activities (e.g. responding to question prompts) (Amulya, 2004; Reynolds, 2011). This enables them to observe and excavate the underlying qualities that make the experience significant.

The evidence of recent research shows that question prompts given by the learner to the tutor in the peer tutoring context have a positive influence. (Wu & Looi, 2011) found that tutor motivation can be built up with tutee questions, which has a significant positive influence in teaching and learning activities. The importance of embedding question prompts into the intelligent learning environment (ILE) design has also been recognized by the researchers in the ILE field (Van der Meij & de Jong, 2011). Question prompts have been extensively used as scaffolds for learner to achieve goals in learning. Positive evidence was found by (Davis, 2000) that question prompts help learners in various aspects, such as problem-solving and knowledge integration (Xie & Bradshaw, 2008). Learners can reflect on their own thoughts by motivating and activating them to analyze and think about the effectiveness of the chosen strategies that would help them to oversee, regulate and control the application or strategic procedure in a specific situation. To present a prompt is usually dependent on the purpose and intention of a specific interposition. The learner is supposed to receive the prompt from the learning tool just at the right time, for example, at the moment that they require assistance or else presenting a prompt in an inappropriate manner will cause cognitive overload (Ifenthaler, 2012; Thillmann, Künsting, Wirth, & Leutner, 2009).

Generally, the difference is made during, before and after the learning sequence presentation. The presentation of the prompt during the sequence of learning is reasonable if the intention is to activate a learner’s monitoring skill in problem-solving activities. However, where the objective is to motivate learners to evaluate certain activities in problem-solving, then presentation after the learning sequence is expedient. It is appropriate to present the prompt before the sequence of problem-solving if one wishes to trigger learner reflection on getting the approach to solve a problem. Another aspect that is crucial is how it can be embedded in order to provide a learner with an optimal scaffold.

According to Davis (2003), there are two categories of reflective prompts; generic and directed prompts in which generic prompts would seem more effective compared to directed prompts because they give the learner autonomy during learning. On the other hand, the directed prompt, asks additional information from the learner to process and introduce a new reflection expert model (see (Davis, 2003)).

2.8.2 Scaffolding

There have been a number of attempts to tackle the difficulties and complexity encountered while learning programming (Apiola, Tedre, & Oroma, 2011; Rum, Nurulain, & Ismail, 2014; Rum & Ismail, 2014; Soloway & Spohrer, 2013). Scaffolding is identified as a critical component in facilitating students' learning (Bickhard, 2013; Feyzi-Behnagh et al., 2014). Scaffolding provides help on an as needed basis to the learner, reducing the assistance as learner competence increases (Kim & Hannafin, 2011). Scaffolding has been defined as support provided for the learner to engage in activities that would instead be beyond their immediate grasp and independent abilities (Davis, 2014; Hannafin, Land, & Oliver, 1999; Vygotskiĭ, 1978). Scaffolds are guides, tools and strategies used by humans, teachers, animated pedagogical agents and computer tutors during learning to develop an understanding beyond their abilities (Deejring, 2015; Graesser, Wiemer-Hastings, Wiemer-Hastings, & Kreuz, 1999).

Strong support for metacognitive skills can be developed by designers and teachers by incorporating the use of metacognitive scaffolds into the curriculum (Hannafin et al., 1999). It has been shown by (Azevedo, Cromley, Winters, Moos, & Greene, 2005; Davis, 2000, 2014; Molenaar, van Boxtel, & Sleegers, 2011) that student attitudes and achievement can be impacted by the use of scaffolds within the learning environment in various situations. According to (Davis, 2014; Molenaar et al., 2011) student success may be affected by metacognitive skills, and they suggest that the mitigation of metacognition skill deficiencies found in students could be realized through the use of strong metacognitive scaffolds.

Metacognitive skills have also been recognized by the school library community as a skill required for students (Brand-Gruwel & Stadtler, 2011; Willer, Eisenberg, & Sadzewicz)). Students that possess strong metacognitive skills yield more positive results in information-seeking behaviors (Choi & Jeong, 2013; Roll, Aleven, McLaren, & Koedinger, 2011). Scaffolding and specific support for desired metacognitive skills can be provided by educators by modeling specific metacognitive activities, such as reflection, self-questioning and strategy revision.

2.8.3 Self-questioning

Self-questioning is an effective way of metacognitive comprehension monitoring strategy for promoting self-directed learning (Joseph, Alber-Morgan, Cullen, & Rouse, 2015). Studies show that questions created by learners are more effective compared to questions given to the learner by someone else. “Have I omitted anything important information?” is an example of a self-question that would help a student in self-directed learning to identify important information. The more self-questions used by students in diverse situations, the more likely it will become a skill that is used unconsciously and automatically as the situation is needed. The performance of the learner before, during and after task performance can be guided through self-questioning. Control over thinking and self-awareness can be improved by self-questioning, thereby improving performance. It can also help the learner to improve the long-term retention of skills and knowledge, as well as the ability to transfer and apply knowledge. Finally, a leaner's motivation and attitude can also be reflected as the result of performance improvement (Brophy, 2013; Rouse, Alber-Morgan, Cullen, & Sawyer, 2014).

2.8.4 Self-asessment and Self-directed

Self-questioning refers to the process through which learners ask and answer questions while reading. This process enables them to understand the text and become independent leaners where they can actively engage through goal-directed learning with organized thinking. In many cases, difficulty in comprehension is always regarded as the failure of the reader to participate actively in the reading process. To increase novice programmers' understanding of the important information in the text as well their motivation, self-questioning is one of the best approaches to practice. (Rum et al., 2014), for example, have designed reflective activities in computer assisted learning systems to encourage self-questioning about the learning experience that lead to self-directed learning. (Merriam, Caffarella, & Baumgartner, 2012) define self-directed as a process in which individuals manage their own learning, identify learning needs, formulate learning goals, implement appropriate learning strategies, apply the required resources and evaluate their learning outcomes

The application of SDL skills offers a way to enable students to 'help themselves' and to adapt to requirements and changes (Francom, 2010). One of Francom's general principles for fostering SDL skills are to provide for both skills – self-directed learning as well as subject matter knowledge. Metacognition is intrinsically part of SDL, and refers to the conscious planning, control and evaluation of one’s own cognitive processes, such as our own thoughts that engage in the learning processes (Sternberg & Frensch, 2014). (Veenman, Van Hout-Wolters, & Afflerbach, 2006) distinguish between metacognitive knowledge and metacognitive skills in which the former refers to a person’s declarative knowledge and the latter comprises a person’s procedural knowledge, for example, how to solve problems. Metacognitive knowledge involves the knowledge of a person, knowledge of a task and knowledge of different strategies (Flavell, 1979). Metacognitive skills, such as monitoring, comprise the ability of an individual to assess his/her state of cognitive activity whereas metacognitive control refers to the ability of an individual to regulate cognitive activity (Miller & Geraci, 2011). Metacognitive control also refers to activities that help students to manage their learning and/or problem-solving (Schraw, 1998) and relates to the planning, monitoring and evaluation of the learning process (Hartman, 2001). Examples of metacognitive skills are motivation, goal orientation, knowledge of one’s own strengths and weaknesses, as well as judgment and beliefs about personal learning (Ertmer & Newby, 1996).

2.8.5 Graphic Organizers

Graphic Organizers, also known as concept maps, mind maps and entity relationship charts, are a pictorial way of organizing information as well as one of versatile reading strategy tools and as metacognitive monitoring instruction. Graphic organizers are used to illustrate conceptual text structures and relationships between text elements and concepts (Stull & Mayer, 2007). According to (Horton, Lovitt, & Bergerud, 1990), a graphic organizer is a text adjunct that is a “visuospatial arrangement of information containing words or statements that are connected graphically to form a meaningful diagram”. Concept maps, knowledge maps, Venn diagrams, causal diagrams, and matrices are just a few of the common graphic organizers seen in textbooks (Stull & Mayer, 2007) and utilized in the classroom. With visual arguments, the organization of text concepts is shown rather than explained, which makes textual relationships more explicit. The comprehension of expository text often depends on the ability of a learner to understand multiple levels of text component relationships and to build a “coherent mental representation of the information being comprehended” (Giora, 1996). Graphic organizers can be used to make the text’s structure and the inferential relationships between and among text elements more explicit (Graesser, Singer, & Trabasso, 1994). During reading a competent learner seeks to build “a coherent cognitive structure that makes sense, by being internally consistent (i.e. coherent internal connections) and consistent with existing knowledge (i.e. coherent external connections)” (Mayer, 2003).

The rhetorical structure of the text is the structure commonly used by writers to organize their ideas and information and the structure used by efficient readers to comprehend text (Chambliss & Calfee, 1998). (Mayer, 2003) describes the most common rhetorical structures found in expository text as generalization, which presents a main idea and describes the supporting evidence; enumeration, which presents a list of facts; sequence, which describes a series of events or steps in a process; classification, which divides information into categories; and compare and contrast, which shows the similarities and differences of two or more items along different dimensions. Figure 2.5 is the definition of the most common types of graphic organizer.

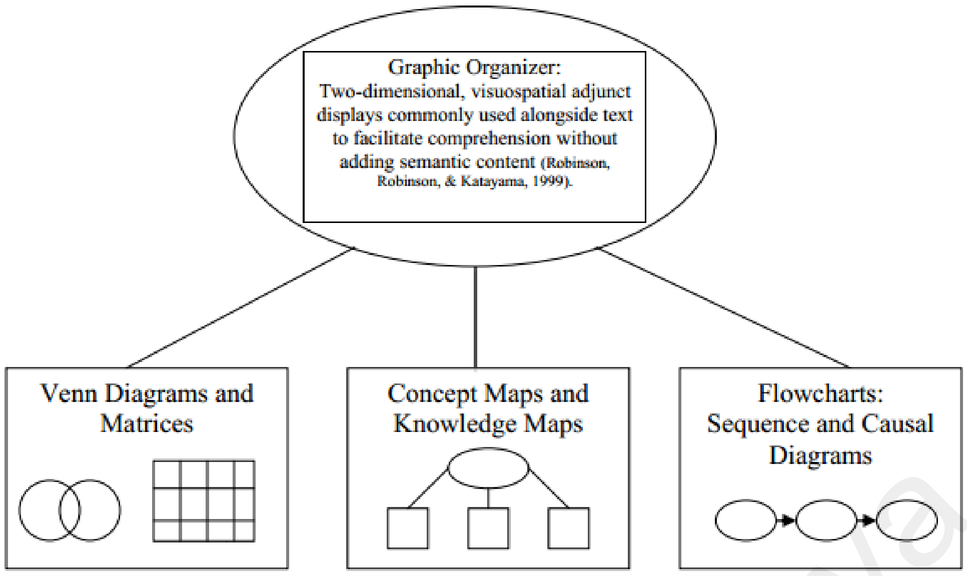

Graphic Organizer:
Two-dimensional, visuospatial adjunct displays commonly used alongside text to facilitate comprehension without adding semantic content (Robinson, Robinson, & Katayama, 1999).
Venn Diagrams and Matrices
Concept Maps and Knowledge Maps
Flowcharts: Sequence and Causal Diagrams

Figure 2.5: Graphic Organizer definition and common graphic organizer types.

2.8.6 Timing

According to Flavell (Flavell, 1979), metacognitive experiences can occur at any time before, during or after cognitive enterprise. In the design of a metacognition support system, we have identified criteria and elements that need to be considered. The design has to be parallel with the goal and objectives of metacognition within the support system. The Tobias and Everson model of metacognition have a limitation of the role within the system, providing the only underlying frame for the development of the proposed system. This section discusses the elements to be integrated within the proposes system framework. Metacognitive experiences take place at any time (during, before, after) the cognitive enterprise. According to (Metallidou & Efklides, 2002), metacognitive experiences can be evoked before, during, or after the task processing is completed and can trigger control decisions. Therefore, the moment in time where the system provides metacognitive instruction is an important consideration. (Zimmerman, 1990) suggested three stages for a better self-regulated learning process:

1. Thoughtful provision beforehand; learners' carefully thinking and preparing in advance before performing task.

2. "Performance control", involves learners consciousness and willpower during the learning process.
3. Self-reflection takes place at the end stage when learners re-examine their performance toward the final goals. Meanwhile concentrating on their strategies during the learning process is efficient for their final results and outcomes (Williams & Hellman, 2004).

Thus, we have put the three stages (before, during and after learning) to present the metacognitive activities within the proposed system as follows:

1. **Before the learning exercise situation**

   Objective: Self-reflection that happens before the learning process leads to the potential to put the learner in the correct condition or frame of mind to perform the task. The possible activities that take place before any attempt to address a new problem or before a new lesson include planning, setting goals, selecting strategies to perform a sequence of operations, identification of potential obstacles and predicting the desired results.

2. **During the learning exercise situation**

   Objective**:** Self-reflection that happens during the learning process can help the student in the self-monitoring process. The possible activities that take place during this stage are the actual cognitive performance, spotting obstacles or errors and knowing how to overcome the obstacles or errors.

3. **After the completion of the learning task**

   Objective: Self-reflection that happens after the learning process is a natural time for the student to reflect on their learning process and performance. The possible activities that take place in this stage are assessing achievement of the goal, inferring the adequacy and accuracy of the results, evaluating the appropriateness of the procedures used, assessing the handling of errors or obstacles, judging the efficiency of the plan and the execution of the plan, etc.

Studies in the cognitive psychology have linked number constructs to metacognition. Critical thinking is one of the constructive elements that has been related to metacognition. Critical thinking varies widely in definitions but the common elements defined by (Ennis, 1985; Ernst* & Monroe, 2004; Paul, 1992) are analyzing arguments, making inferences through the deductive reasoning or inductive reasoning, evaluating or judging, solving problems or making decisions. They suggest that instruction should designate a process of groundwork in general principles of critical thinking, as well as practice in applying critical thinking skills in the context of particular domains. An instructional approach, called Cognitive Strategy Instruction (CSI), stresses that the development of the thinking processes and skills as a means to enhance learning is an important criterion to enable the learner to be more self-reliant, strategic, productive and flexible in their learning effort (Scheid, 1993). Underlying CSI is the presumption that cognitive strategies are identifiable (Halpern, 2002) and that learning success has been connected with these strategies (Garner, 1990). With metacognition, the learner can benefit from instruction (Carr et al., 1989; Zile-Tamsen & Marie, 1996) and impel the maintenance and the use of cognitive strategies. While there are a few existing metacognitive instruction approaches, the most effective one is preparing the learner with the knowledge of the cognitive processes as well as metacognitive strategies. In the development of metacognitive regulation, exposure or practice in using both cognitive and metacognitive strategies and evaluating the outcomes of their efforts are the key contributors. Simply preparing knowledge without experience or the other way around does not appear to be adequate for the development of metacognitive control (Livingston, 1996).

2.9 Existing Support Learning Tools for Novice Programmers

E-learning has a lot of potential as a metacognitive tool. For instance, through their ability to record interactions with users, they can become powerful reflection tools. Having captured the actions of the student carrying out a task, these can be played back to them in an abstract and structured way. This will help the student to become aware of the processes and help them improve performance on the task in question through reflection on the how's and why's of the chosen problem-solving paths. As collaborative learning devices, they can be programmed to support group planning, monitoring and evaluation of the learning process. Students in a small learning group can, for example, look back over their solution paths and compare them with other members of the group. This should trigger reflection on which changes could be improved (Boud, Keogh, & Walker, 2013). Another interesting possibility is that of simulated "Learning Companions" acting as peers who encourage the student to reflect and articulate their actions (Scott, 2014) Despite this potential, to date, the majority of computer-based learning environments have focused on supporting students by developing domain-related skills and knowledge. Some attempts have been made to incorporate metacognitive components, mostly in the form of embedded reflection on the learning task or processes. In addition, a very small number of systems have included explicit metacognition training as their main target. Detecting, tracing, modelling, and fostering students metacognitive and self-regulatory behaviors during learning in ILEs is a research challenge that is almost ignored. Nevertheless, the examples from existing literature show the prospects for this important area.

Finding similar works that specifically focus on support systems in learning computer programming metacognitively and using the Semantic Web as the underlying technology for building the support tool could not be located. However, a few studies were identified that were similar in terms of providing a support tool in learning programming. ‘Gidget’ (Lee & Ko, 2011), for example, provides features as follows:

- A game where the eponymous robot protagonist is cast as a fallible character that blames itself for not being able to correctly write code to complete its mission.
- Players learn programming by working with ‘Gidget’ to debug its problematic code. In a two-condition controlled experiment, we manipulated ‘Gidget’ level of personification in communication style, sound effects, and image.

ANNET (Liffick B.W. & Aiken R., 1996) using the cognitive model approach to develop an automated method of annotating example programs in such a way that students can access information about any aspect of the example that was preventing them from fully understanding it. This model is a synthesis of the interactive language model proposed by Foley, van Dam et al. (Foley, Van Dam, Feiner, Hughes, & Phillips, 1994) and the concept of programming plans as described by Soloway, Bonar, and others (Soloway, 1984). The result is a definition of a cognitive model containing five levels of knowledge within the domain of programming: Lexical, Syntactic, Semantic, Schematic, and Conceptual, The Lexical and Syntactic levels are self-explanatory. Figure 2.6 presents the hierarchical nature of ANNET from the generalized and specialized perspective.

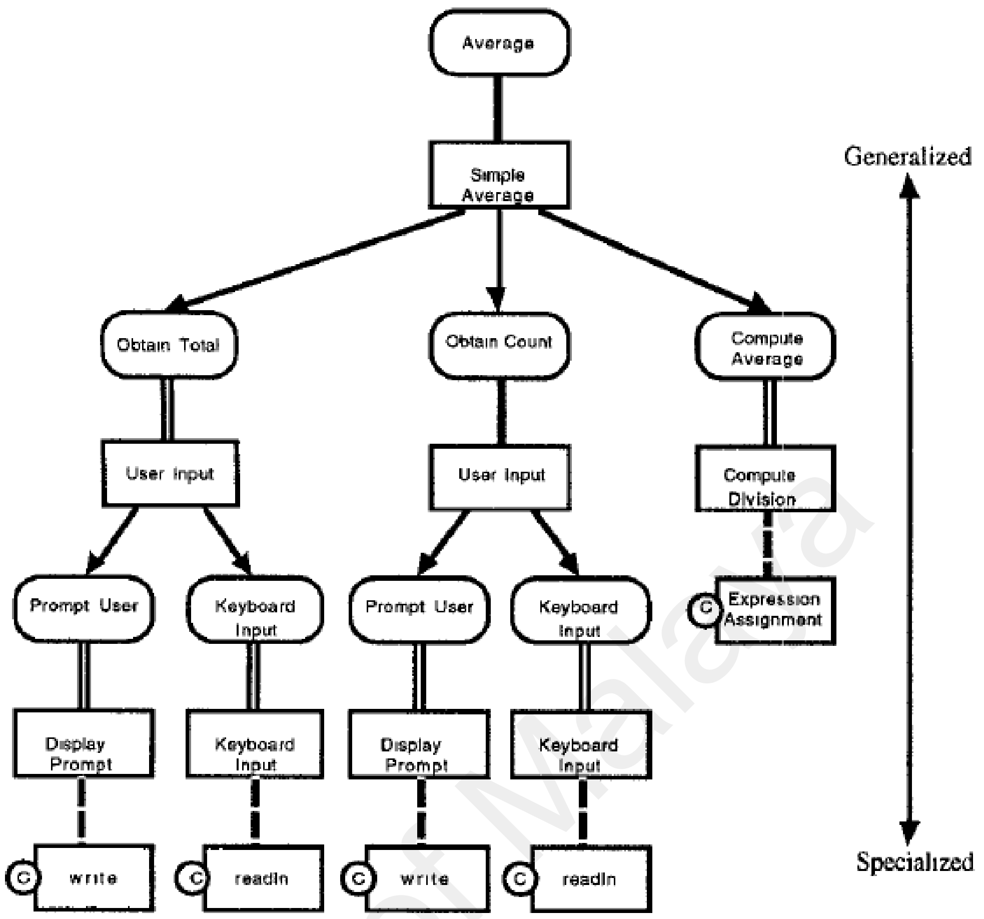


Figure 2.6: The hierarchical nature of ANNET (Liffick B.W. & Aiken R., 1996)

CALMS (Thota & Whitfield, 2009) (Figure 2.9) is a standard LMS that can be customized by instructors according to their preferences by integrating computer science content with plug-in modules of learning software developed by themselves or others. The pedagogic foundation for the introductory programming course design including learning theories, instructional process and learning taxonomies is discussed along with the choice of educational media and the organization of the learning content. It is designed to be integrated dynamic programming visualization with Moodle that was guided by the need to have a flexible, adaptable and collaborative learning environment. Some of the programming related activities and resources are shown in Figure 2.7.

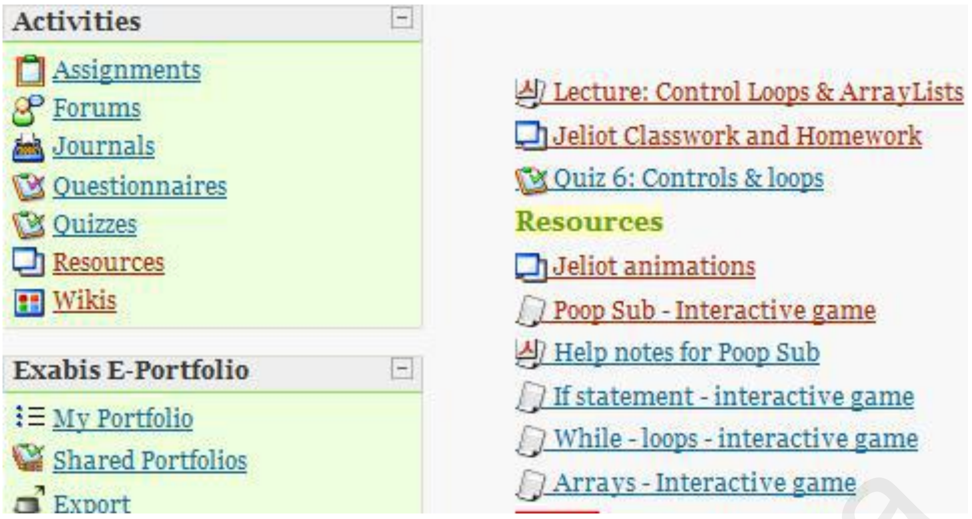
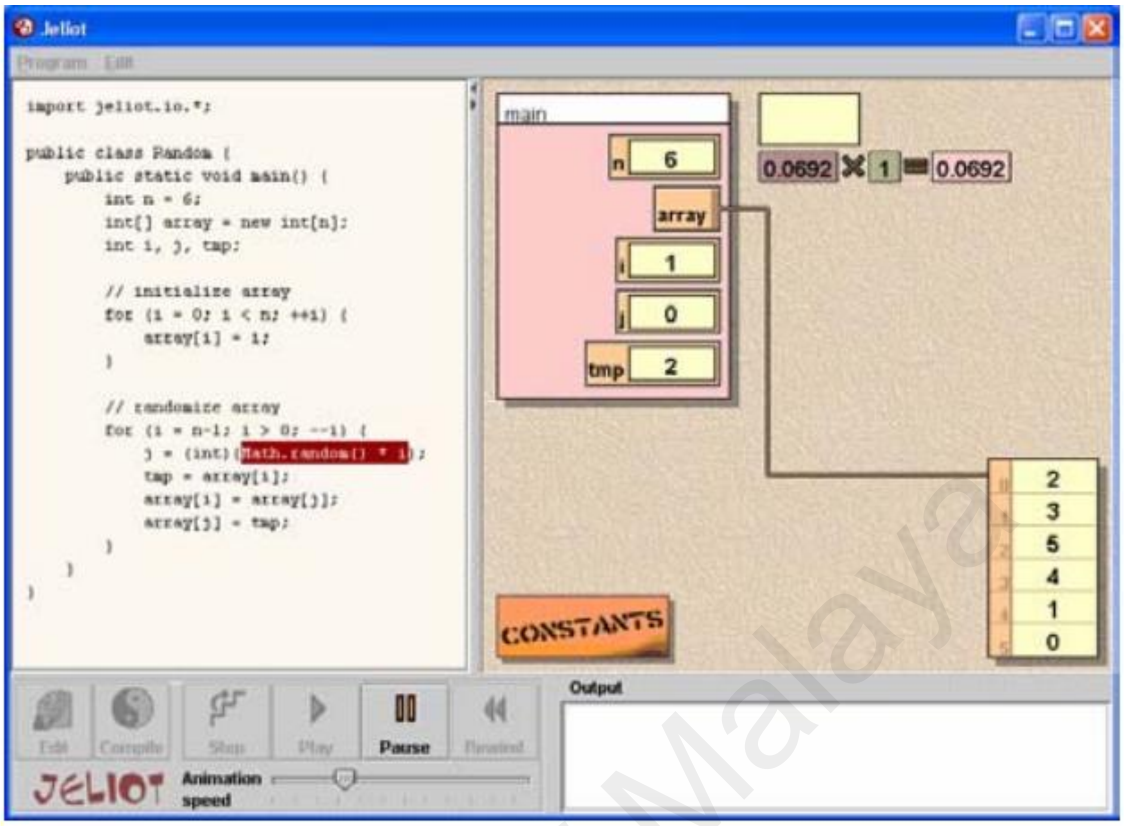


Figure 2.7: Moodle activities and resources for programming course (Thota & Whitfield, 2009)

Jeliot 3 (Moreno, Myller, Sutinen, & Ben-Ari, 2004) (see Figure 2.8) is designed to aid novice students to learn procedural and object-oriented programming. The key feature of Jeliot is the fully or semi-automatic visualization of the data and control flows. The development process of Jeliot has been research-oriented, meaning that all the different versions have had their own research agenda arising from the design of the previous version.

JAVAVIS (Oechsle & Schmitt, 2002) is a system developed to help students understand what is happening in a Java program during execution. The system uses the Java Debug Interface (JDI). It visualizes the state of the program and its changes during execution. The system is not meant for novices, because the visualization it produces assumes that students are familiar with UML and the basics of programming. However, this kind of system could be very useful for advanced courses in programming. Table 2.4 presents a summary of the features provided by each support tool.



```
import jeliot.io.*;

public class Random {
    public static void main() {
        int n = 6;
        int[] array = new int[n];
        int i, j, tmp;

        // initialize array
        for (i = 0; i < n; ++i) {
            array[i] = i;
        }

        // randomize array
        for (i = n-1; i > 0; --i) {
            j = (int)(Math.random() * i);
            tmp = array[i];
            array[i] = array[j];
            array[j] = tmp;
        }
    }
}
```

Figure 2.5: User interface of Jelliot 3

Table 2.4: Summary of support tool features

Support tool	**Scaffolding**	**Modeling**	**Self-Assessment**	**Graphic Organizer**	**Self-Directed**
Gidget (Lee & Ko, 2011)	√			√	√
CALMS (Liffick B.W. & Aiken R., 1996)	√	√			
Jeliot 3 (Moreno et al., 2004)	√	√		√	√
JAVAVIS (Oechsle & Schmitt, 2002)	√			√	
ANNET (Liffick B.W. & Aiken R., 1996)	√	√			√

While there are several approaches to metacognitive instruction, the most effective ones involve a mixture of theory and practice. The learner must be given some knowledge of cognitive processes and strategies (that will be used as metacognitive knowledge), as well as opportunities to practice both cognitive and metacognitive strategies; evaluation of the outcome of their efforts is also important for the development of metacognitive regulation (Brown, 1987; White et al., 1999). Simply providing knowledge without experience or vice versa does not seem to be sufficient for the development of metacognitive control. Hence, it is necessary to design metacognitive activities that can be embedded into instructional contexts. In this sense, Lin (2001) suggests that the design of such metacognitive activities should focus on both cognitive and social aspects of student development, including strategy training and the creation of a supportive social environment for the teaching of two kinds of content: knowledge about a specific domain and knowledge about the self-as-learner. She also affirms that this balanced approach is a theoretical and practical challenge for teachers and researchers.

Researchers have identified strategies that teachers can use to promote metacognition in the classroom. An interesting set of activities and tasks that can be adapted and used by teachers in different situations is proposed by Angelo and Cross (1993). It is worth noting that sometimes students apply one or more techniques that are ineffective. Hence, it is important to evaluate students' metacognitive abilities and target instruction to the development of more effective and adequate general learning strategies. Some of the example of metacognitve activities that can be embedded in instructional program are planning, goal setting and assessing knowledge monitoring.

2.10 Semantic Web as the Underlying Technology for E-learning

Many E-learning applications are highly monolithic and seriously lacking in flexibility. The kind of intelligent computer support enabled by Semantic Web descriptions, such as software agents and self-describing systems, is not taken into account in the design. The key property of the Semantic Web architecture, (i.e. common-shared-meaning and machine-process able metadata), which is enabled by a set of suitable agents, establishes a powerful approach to satisfy the E-learning requirements: efficient, just-in-time and task relevant learning. Knowledge based learning is semantically annotated and for a new learning demand it may be easily combined in a new learning course. In fact, the Semantic Web can be exploited as a very suitable platform for implementing an E-learning system, because it provides all means for eLearning: ontology development, ontology-based annotation of learning materials, their composition in learning courses and proactive delivery of the learning materials through E-learning portals. More details about the E-learning scenario will be given in the last section. In the following (Table 2.5), a summary view of the possibility to use the Semantic Web for realizing the E-learning requirements is presented.

Table 2.5: Semantic Web as underlying technology for E-learning: the benefit (Drucker, 2000)

Dimensions	**Training**	**eLearning**
Delivery	Pull – Student determines the agenda	Knowledge items (learning materials) are distributed on the Web, but they are linked to a commonly agreed ontology. This enables the construction of a user-specific course, by semantic querying for topics of interest
Access	Reactionary – Responds to problem at hand	Software agents on the Semantic Web may use a commonly agreed service language, which enables co-ordination between agents and proactive delivery of learning materials in the context of actual problems. The vision is that each user has his own personalized agent that communicates with other agents.
Symmetry	Non-linear – Allows direct access to knowledge in whatever sequence makes sense to the situation at hand	User can describe the situation at hand (goal of learning, previous knowledge,...) and perform semantic querying for the suitable learning material. The user profile is also accounted for. Access to knowledge can be expanded by semantically defined navigation.
Modality	Continuous – Learning runs in parallel to business tasks and never stops	Active delivery of information (based on personalized agents) creates a dynamic learning environment that is integrated in the business processes.
Authority	Distributed – Contents come from the	The Semantic Web will be as decentralized as possible. This form of

	interaction of the participants and the educators	interaction enables an effective co-operative content management.
Personalization	Personalized – Contents determined by the individual user's needs and aims to satisfy the needs of every user	A user (using its personalized agent) searches for learning material customized for her/his needs. The ontology is the link between user needs and characteristics of the learning material
Adaptively	Dynamic – Content changes constantly through user input, experiences, new practices, business rules and heuristics	The Semantic Web enables the use of distributed knowledge provided in various forms, enabled by semantic annotation of content. Distributed nature of the Semantic Web enables continuous improvement of the learning materials

2.10.1 Why semantic Web for the MSSNP?

Semantic Web is the enabling technology facilitates the process of distinguishing the ambiguity lies between cognition and metacognition. With this technology information that relates to both of the skills can be expressed in precise form that can be interpreted by machine and ready for software agents to process, share that would enable the application to interoperate on the semantic as well as syntactic level. In the literature, the cognitive and metacognitive functions are often used interchangeably (Frith, 2012; Jones & Idol, 2013; Markova & Legerstee, 2013; Mayer, 2011; Özsoy, 2011; Schellenberg, Negishi, & Eggen, 2011; Son & Simon, 2012; Vohs et al., 2014). The ambiguity mainly comes from the following three reasons: (1) it is difficult to distinguish metacognition from cognition; (2) metacognition has been used to refer to two distinct areas of research: knowledge about cognition and regulation of cognition; and (3) there are four historical roots to the inquiry of metacognition (Brown Ann, 1987). With this ambiguous definition of "metacognition", we cannot answer the crucial questions concerning existing learning strategies or systems: what they have supported, or not; what is difficult for them to support; why it is difficult; and essentially what the distinction between cognition and metacognition is. In order to answer these questions, the ontology is used to explicitly formalize the specification of the terms in cognitive and metacognitive and relations among them. We first should clarify how many concepts are subsumed under the term metacognition and how each of these concepts depend upon each other. The constructed of the set of vocabulary contains the important concept and classess and knowledge and semantic are coded into the set.

This clarification through the use of ontology (RDF standard model for data interchange on the Web) of the difference between the cognitive and metacognitive skills enables us to specifying the goals for learning strategies, and systems to support the development of learners' metacognition; what and why it is difficult to achieve each of these goals; and how to achieve each of the goals using strategies within the support systems.

2.10.2 Semantic Web Technologies

(Lee, Hendler, & Lassila, 2001) said that "*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in co-operation*". The Semantic Web is the current technology that enables machines to interpret data published in the machines into an interpretable form on the Web to more meaningful content. Much work has been done to improve the Web representation from HTLM to semantic based language, such as extended markup language XML, Resource Description Framework RDF, RDF schema (RDFS) and Web Ontology Language (OWL).

The Semantic Web Stack (illustrated in Figure 2.9) of the hierarchy of languages, is where each layer exploits and uses the capabilities of the layers below. This shows how technologies that are standardized for the Semantic Web are organized to make the

Semantic Web possible. It also shows how the Semantic Web is an extension (not replacement) of the classical hypertext web.

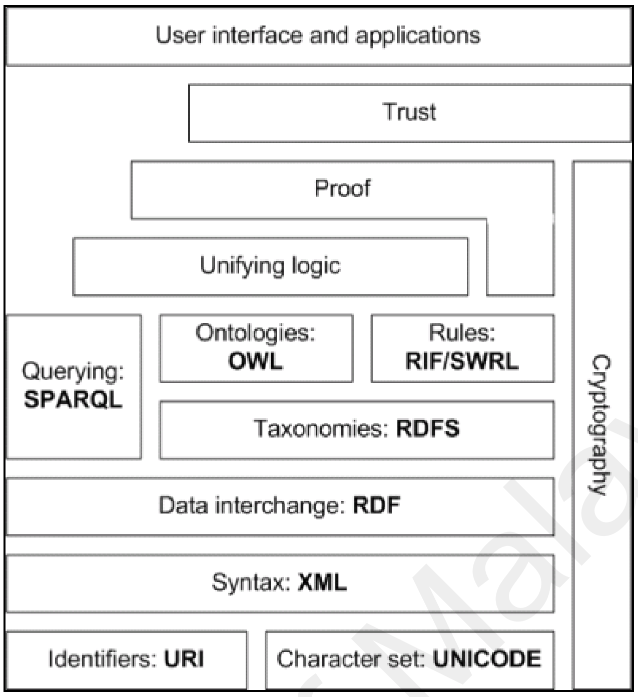


Figure 2.6: Semantic Web Stack (Berners-Lee, Fischetti, & Foreword By-Dertouzos, 2000)

2.10.3 Layers of the Semantic Web

The main task of the Semantic Web is to express meaning. To achieve this, representation structures (several layers) are required (see Figure 2.10), for which the basic layers are as follows:

- XML layer (represents data structure)
- RDF layer (represents the data meaning)
- Ontology layer (represents formal meaningful of data)
- Logic layer (intelligent reasoning)

The real power behind the Semantic Web technology is realized when the content of the Web from diverse source exchanges processes the information with machine agents or other humans. As more machine readable web content, as well as automated services become available, there will be a drastic increase in the effectiveness of the Semantic Web. To achieve this, the "proofs" exchange is required for inter agent communication. There are two technologies that are already in place for the development of the Semantic Web; the eXtensible Markup Language (XML) and the Resource Description Framework (RDF).

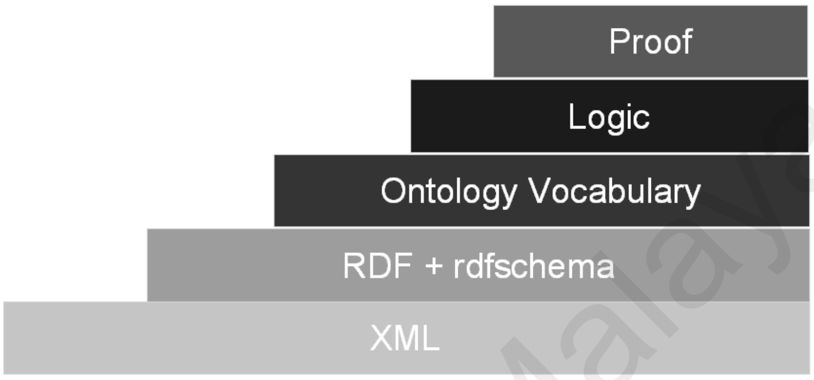


Figure 2.10: Layer of Semantic Web

With XML, programmers can create their own tags to annotate sections of text as well as on a page in sophisticated ways. It can be used by others who need to know the purpose of the tag. In short, XML gives flexibility to the user to put arbitrary structure to their documents without the meaning of the structure (Erdmann & Studer 2000). With the "semantic" mark up and tags, the meaning of XML-documents is intuitively clearly defined. However, the tag names alone do not provide semantics as computers do not have intuition. The RDF provides a means for adding semantics to a document. With RDF, structured metadata can be encoded, exchanged and reused. Primarily, information is machine understandable, which is stored in triple based form RDF statements. Intelligent agents, search engines, browser, information brokers and humans can understand the information as well as use that semantic information. The implementation of RDF is independent serialized in XML. Semantic annotation is a process of adding semantic information to the Web (Handschuh et al., 2001). The combination of RDF and RDFS offers primitive modeling that gives the flexibility of expansion according to the demand. RDFS can be used to express the relationship between object and classes. In general, there is a lack of a formal semantic for RDF(S), for its modeling primitives that make interpretation an error-prone process. Realizing this, the viz ontologies, the third component of Semantic Web component provide the solution.

2.10.4 Hypertext Web Technologies

The basis technologies of the Semantic Web are provided in the bottom layers (Berners-Lee et al., 2000).

- Uniform Resource Identifier (URI) provides a means to the resources of the Semantic Web that are uniquely identified to allow provable manipulation with resources in the top layer
- Unicode – server to represent and manipulate text in many languages. Semantic Web should also help to bridge documents in different human languages
- XML – markup language that enables the creation of documents composed of structured data. Semantic web gives meaning (semantics) to structured data.
- XML Namespaces

2.10.5 Standardized Semantic Web technologies

Middle layers contain technologies standardized by W3C to enable building Semantic Web applications (Berners-Lee et al., 2000):

- Resource Description Framework (RDF) is a framework for creating statements in a form of so-called triples. It enables the representation of information about resources in the form of a graph.
- RDF Schema (RDFS) provides basic vocabulary for RDF. Using RDFS it is for example possible to create hierarchies of classes and properties.
- Web Ontology Language (OWL) Extends RDFS by adding more advanced constructs to describe the semantics of DRF statements.

- SPARQL is a RDF query language can be used to query any RDF-based data (including RDFS and OWL) to retrieve information for Semantic Web applications.

2.10.6 Unrealized Semantic Web Technologies

Top layers contain technologies that are not yet standardized or contain just ideas that should be implemented in order to realize the Semantic Web:

- RIF or SWRL will bring the support of the rules. This is important for example to allow describing relations that cannot be directly described using description logic used in OWL.
- Cryptography is important to ensure and verify that Semantic Web statements are coming from a trusted source. This can be achieved by the appropriate digital signature of RDF statements.
- Trust to derive statements will be supported by (a) verifying that the premises come from trusted sources and by (b) relying on formal logic during deriving new information.
- The user interface is the final layer that enables humans to use Semantic Web applications.

2.10.7 RDF and RDFS

RDF is a data model of metadata that provides universal data translation into XML format. The idea of a RDF data model is to make statements about web resources in the triple based expression (subject-object-predicate). In RDF terminology, these expressions are known as triples. RDF representations are depicted as a directed labeled graph, as illustrated in Figure 2.13. The subject in RDF can be either a blank node or a

Uniform Resourcer Identifier (URI). Both URI and blank node denode resources. Anonymous resources is the term used to indicate blank node, which are not directly identifiable from the statement of RDF.

This is predicated on the fact that it is also a URI to denote resources as well as represent the relationship. Whereas the object can also be a blank node, a Unicode string literal or URI. In the triple data store, the subject is linked to other resources through a property link. A statement is another term for triple. In Figure 2.11, the RDF triples, subject-predicate-object, are used to represent information that can be read as ‘Dora Smith has homepage http://www.sample.org/~dora’.

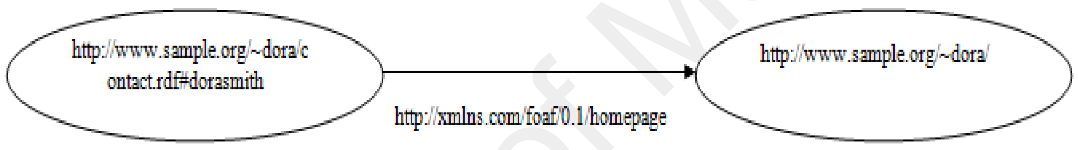

http://www.sample.org/~dora/contact.rdf#dorasmith
http://xmlns.com/foaf/0.1/homepage
http://www.sample.org/~dora/

Figure 2.7: Example RDF statement graphical representation

2.11 Ontology

Behind the Semantic Web technology is Ontology. Ontologies are the backbone of the Semantic Web. There have been many attempts to define what constitutes ontology, but according to (Gruber, 1993) *“An Ontology is an explicit specification of a conceptualization”.* A conceptualization in this context means an abstract model of some aspects of the world, taking the form of a definition of the properties of important concepts and relationships. An explicit specification means that the model should be specified in some unambiguous language, making it amenable to processing by machines as well as by humans. Furthermore (Guarino, 1998), suggests the opportunity to develop different kinds of ontology according to their generality, as shown in Figure 2.12 (for a more detailed discussion see (Van Heijst, Schreiber, & Wielinga, 1997)).

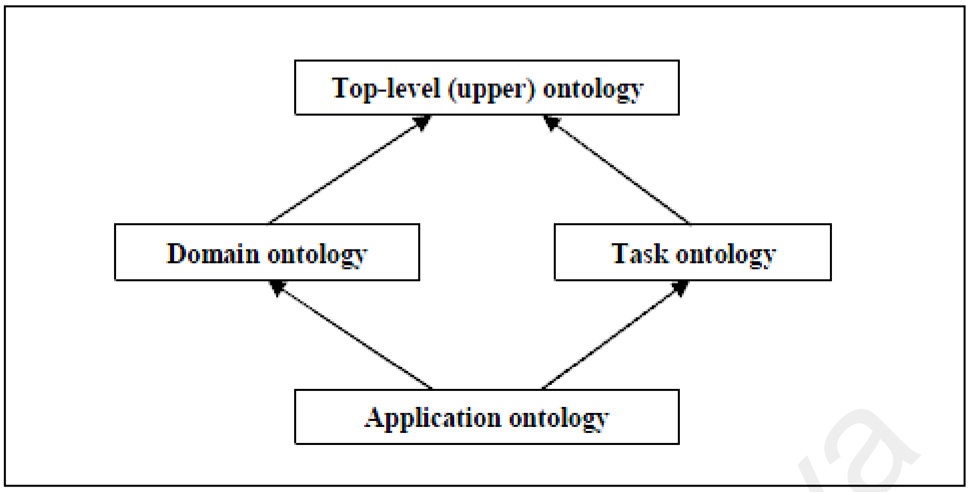


Figure 2.8: Kinds of Ontology (Guarino, 1998)

- Top-level (upper) Ontologies – this type of ontology describes very general concepts of the ontology, such as space, time, matter, object, event, etc., which are independent of a particular problem or domain; it therefore seems reasonable, at least in theory, to have unified top-level ontologies for large communities of users.
- Domain Ontologies and Task Ontologies – describe, respectively, the vocabulary related to generic domain (like medicine, or automobiles) or a generic task or activity (like diagnosing or selling), by specializing the terms introduced in the top-level ontology
- Application Ontologies – application ontology that consists of representations of defined classes.

2.11.1 Ontology Construction

In traditional learning, instructors play an important role intermediate between learning material and learner but it is completely different in the learning scenario in E-learning where the learning material and learners are no longer controlled by the instructors. In recent Web-based learning, a broad variety of learning materials are available for learners. The new generation of Web technology called the Semantic Web makes it possible to express the information in a precise way that is understood by machines and ready to process by a software agent. Compared to traditional learning in which the instructor plays the intermediate role between the learner and the learning material, the learning scenario in E-learning is completely different: instructors no longer control the delivery of the material and learners have the possibility of combining learning material in courses on their own. Recent advances in Web-based learning technology provide a broad variety of learning materials available to people.

The Semantic Web is the new-generation Web that makes it possible to express information in a precise, machine interpretable form, ready for software agents to process, share, and reuse it, as well as to understand what the terms describing the data mean. Thus, ontology plays a vital role in ensuring the homogeneity of information by allowing them to share the semantics. In this research work, an ontology is identified as domain specific ontology in learning introductory programming metacognitively. The motivation behind ontology construction is for a shared understanding of metacognitive skill and facilitating its development as well as to organize activities in cognitive skill and metacognitive skill. According to (Wache Holger et al., 2001; Wache H. et al., 2001), ontologies can be constructed using three different approaches – single ontology, multiple ontology and hybrid ontology (Wache H. et al., 2001). These approaches help in providing the integration task to describe the semantics of information sources. In general, ontology construction could be done in three ways (Ismail, Yaacob, Kareem, & Halim):

- Manual whereby ontology is constructed manually
- Semi-automatic whereby human intervention is needed during the ontology process

- Fully automatic whereby the system takes care of the complete construction

The single ontology approach (as shown in Figure 2.13) is the most simple approach of ontology construction, which shares terminology as well as the vocabulary for specifying the semantics. However, there is a limitation of this approach in that it does not provide a solution for the integration of information. In multiple ontology approach (see Figure 2.14), information source is described by its own local by its own. The hybrid ontology approach (see Figure 2.15) is the combination of single and multiple approaches (Wache Holger et al., 2001).

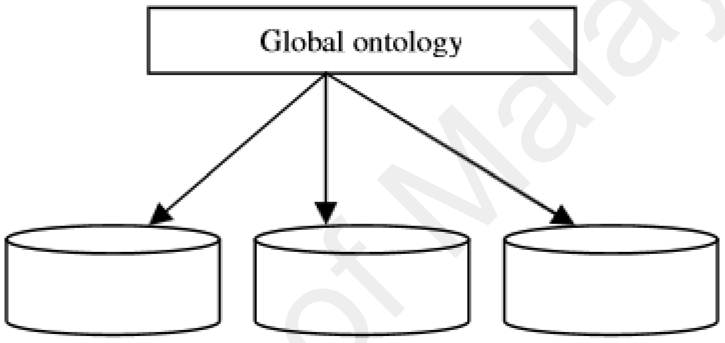


Figure 2.13: Single Ontology Approach

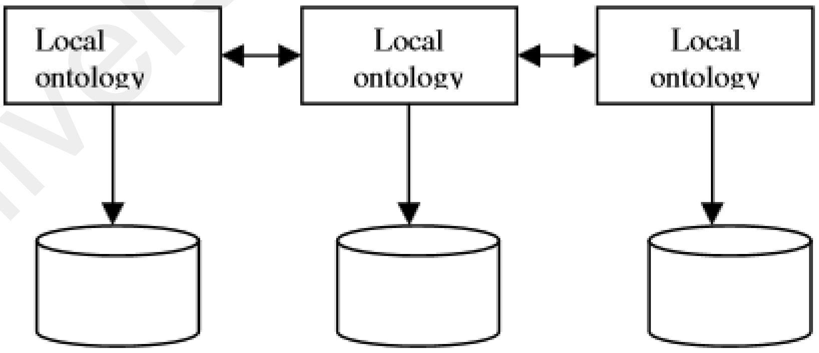


Figure 2.14: Multiple Ontology Approach

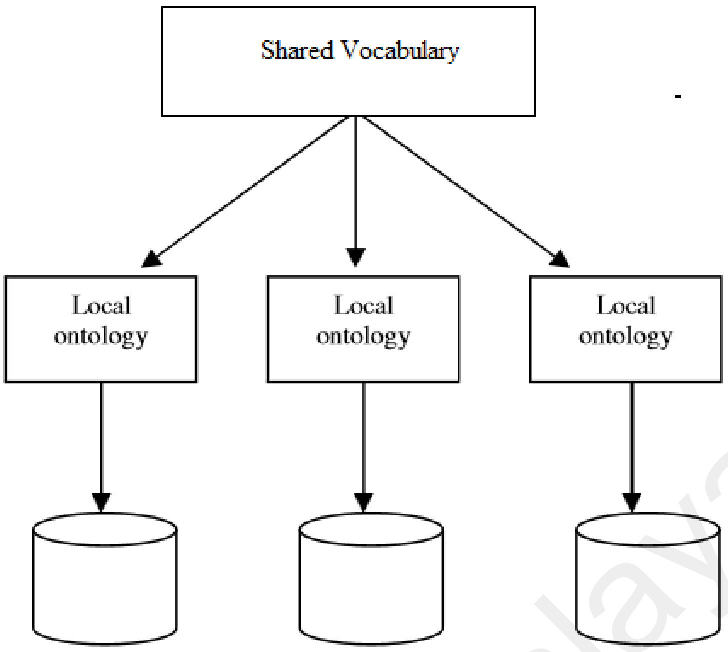


Figure 2.15: Hybrid approach

Ontology construction is not an isolation process but an iterative process that involves the following steps (Ismail et al.):

- Design – specification of the scope and objective of the ontology as well as identification of the relationship among objects.
- Develop – make decision as to whether to use the existing ontology or to construct the ontology from scratch
- Integrate – combine the built ontology with the existing one.
- Validate – verification of the constructed ontology by using automated tools or by experts
- Iterate – repeat the above process as well as incorporate the changes made from the validation process.

(Rum et al., 2014) described the seven steps in building the Ontology. The first step is the scope and boundary definition. One of the ways of determining the scope of the ontology is by sketching a list of questions and competency questions (Grüninger &

Fox, 1995) concerning how to achieve its objectives. Later, the litmus test of using these questions is used to determine whether the ontology can provide the answer for the competency question or not. This is about making the content of E-learning more understandable by machines (McIlraith, Son, & Zeng, 2001), and, furthermore, adaptively in the context of user input and experience. In the second step, the reuse consideration, reusing existing sources may be a requirement for a system that needs to communicate with other applications that have already committed to a specific vocabulary that is controlled by the organization, field or domain, etc. The third step involves the class design enumerates terms by listing down all the terms that we intend to use to explain things to a user. Taking into consideration the properties of the terms; for example, in this study, important metacognitive learning related terms will include the student, syllabus, performance, knowledge monitoring, strategy and so on.

First and foremost, it is important to comprehensively list down all the related terms, concepts or classes. The next step is to develop the association, class hierarchy between classes and defined properties. In step four, the "Identification and Representation of Taxonomic Relationship" aims at discovering taxonomic relationships through the application of heuristic patterns and at representing them in an ontology specification language. There are three approaches to generalizing and specializing the key concept, that is, top-down, bottom-up and a combination of these two approaches. The sixth step involves property identification. In order to answer the competency questions, classes alone will not be sufficient to provide the information. Once the classes have been identified, the concept's internal structure must be described. The anomalies check is the final step of the construction process to check the inconsistency of the ontology design.

2.11.2 Ontology Development Tools

In ontology life cycle, creation, implementation, population and maintenance of ontologies can be made using the ontology tools (Polikoff, 2003). Various types of knowledge management, such as retrieval, knowledge sharing can be provided by ontology (Pundit & Bishr, 1999). The most popular definition of ontology is "shared knowledge specification" (Waterson & Preece, 1999). In the context of knowledge-management system, ontology can be referred to as knowledge classification. Ontology based search engines are different to traditional keyword based search engines in terms of semantic matching and the functionality in that they have the capability to do retrieval more efficiently compared to the traditional methods. The process of ontology construction is a laborious task and very time consuming. According to Gunther (1998), XML language is not suitable to describe the interrelationships of resources as well as machine-understandable documents. Therefore, the use of RDF, RDFS, DAML+OIL, and OWL have been recommended by W3C. Since then, many tools have been developed for implementing the metadata of ontologies by using these languages. Table 2.6 presents the general description of the ontology tools and the prominent ontology construction tools available as open source or software license. Table 2.7 ontology software architecture tools and the evolution.

Table 2.6: General description of Development tools

Development Tool	**Tool Description**	**Developers**	**Availibility**
Protégé (Noy et al., 2001)	Ontology developers can perform knowledge management task (navigate, manipulate and manage ontology). Provide Tree control for quick navigation through class hierarchy	SMI (Standford University)	Open Source
OntoEdit (Sure, Angele, & Staab, 2003)	Construct a collection of relatively small content rich theory ontologies, and to compose them together to form larger ontological	Ontoprise	Software License

	structures (Swartout et al., 1996)		
Apollo (Koss, 2002)	Construct a collection of relatively small content ontologies. Ontology is represented in hierarchical format	KMI (Open University)	Open Source
SWOOP (Kalyanpur, 2004)(see Figure 2.18)	Consist of OWL validation and syntax presentation views (Abstract Syntax, N3, etc.). Provide reasoning feature (Pallet and RDFS-like), support OWL Inference engine as well as provide a multiple ontology environment	MND (University of Maryland)	Open Source

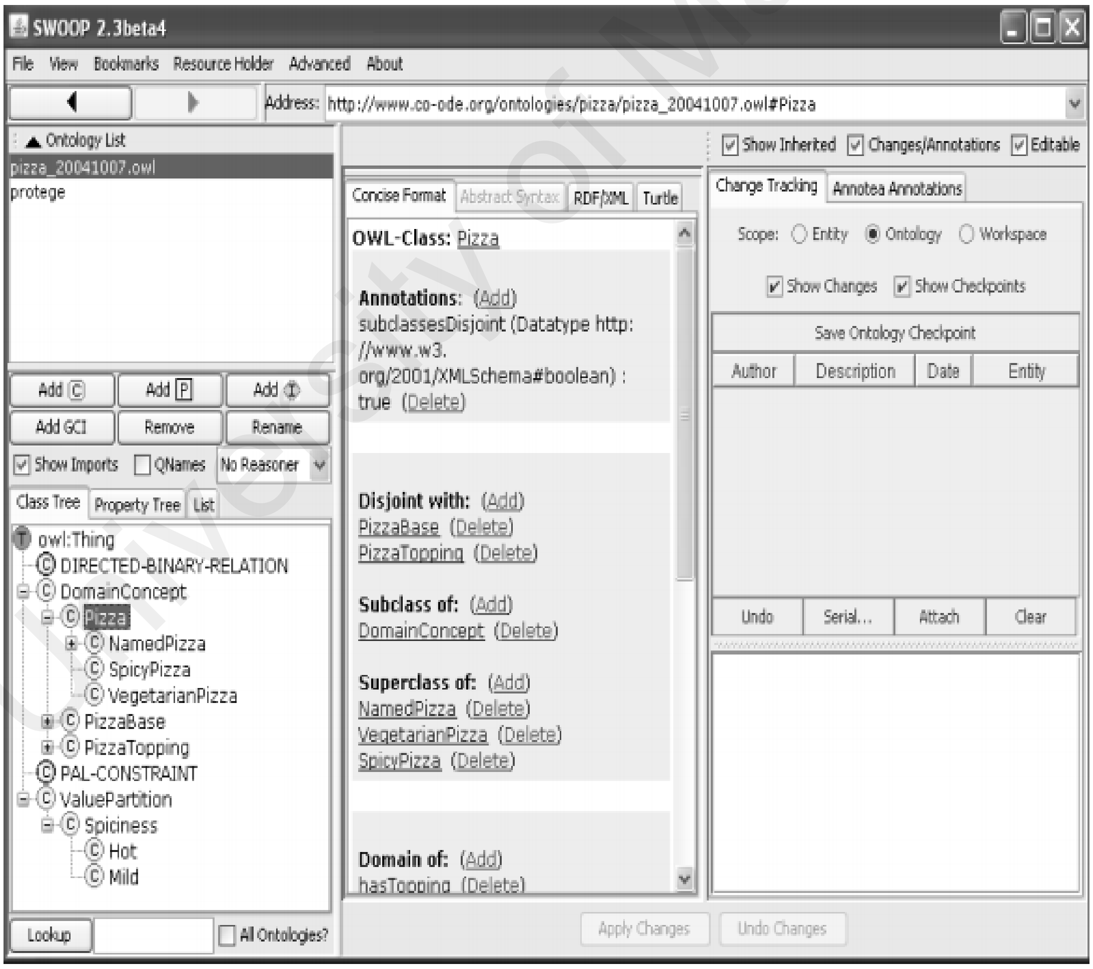


Figure 2.9: SWOOP screenshot

Table 2.7: Software architecture and tool evolution

Feature	**Apollo**	**OntoEdit**	**Protégé**	**Swoop**	**TopBraid Composer**
Semantic web architecture	Standalone	Eclipse client/server	Standalone and Client Server	Web-based and Client Server	Standalone Eclipse plug-in
Extensibility	Plug-ins	Plug-ins	Plug-ins	Plug-ins	Plug-in
Backup Management	No	No	No	No	Yes
Ontology Storage	Files	DBMS	File and DBMS (JDBC)	As HTML Models	DBMS

2.12 Summary

From the review of the literature, it is clear that metacognition is a multi-aspect research topic and providing training for metacognition is a challenging task. This leads us to conclude that metacognition learning environments involve more than just simple learning activities. The three prominent models of metacognition (Flavell’s model, Brown’s model, Tobias and Everson model) give an insight of the characteristic that must be incorporated in the metacognitive support learning environment. A review of the existing literature shows the following weaknesses in the existing support tools:

1. All of the discussed support tools focus on the cognitive development without considering the metacognitive development.
2. Some of the systems are not meant for novices, and only suitable for advanced programming learning.

With the intention to improve the observable domain, it is necessary to synchronize the metacognitive message to the domain and seamlessly blend it into teaching and learning. The literature has helped in identification of the importance of metacognitive skills in problem-solving. However, converting these findings into a computerized environment poses a major challenge, especially without the ability of the lecturer to adapt metacognitive scaffolding in terms of time and the amount of individual needs. Although many attempts have been made by researchers for helping students to develop their metacognition skills and knowledge, especially in the areas of Intelligent Tutoring Systems (ITS), only a few achieved significant results as well as making metacognition the main goal. The outline of the necessary steps that are ideal for an ideal environment is presented in the following chapter. The studies also revealed that Semantic Web technology is the most suitable solution to develop the prototype system as it allows one to differentiate the cognitive and metacognitive elements and activities.

CHAPTER 3: REALIZATION OF A SUPPORT ENVIRONMENT FOR NOVICE PROGRAMMERS

The research flow of this research work is presented in this chapter. It shows the steps taken to achieve the aim to identify the methods of supporting teaching and learning as portrayed in Figure 3.1. The methods involved in this study were observational study to understand how a pedagogical approach assists student during the computer programming class; a survey using a questionnaire to investigate the effect of metacognition on the learning success of computer programming at university; interviews with the novice programmers with the objective of understanding their learning behavior; interviews with the expert lecturers to explore the metacognitive implementation and metacognitive awareness in teaching and learning the computer programming course at universities; identification of the metacognitive learning methods based on the literature study; designing and implementation of the proposed support system for novice programmers, and, finally, the evaluation of the usability of the prototype system by the user.

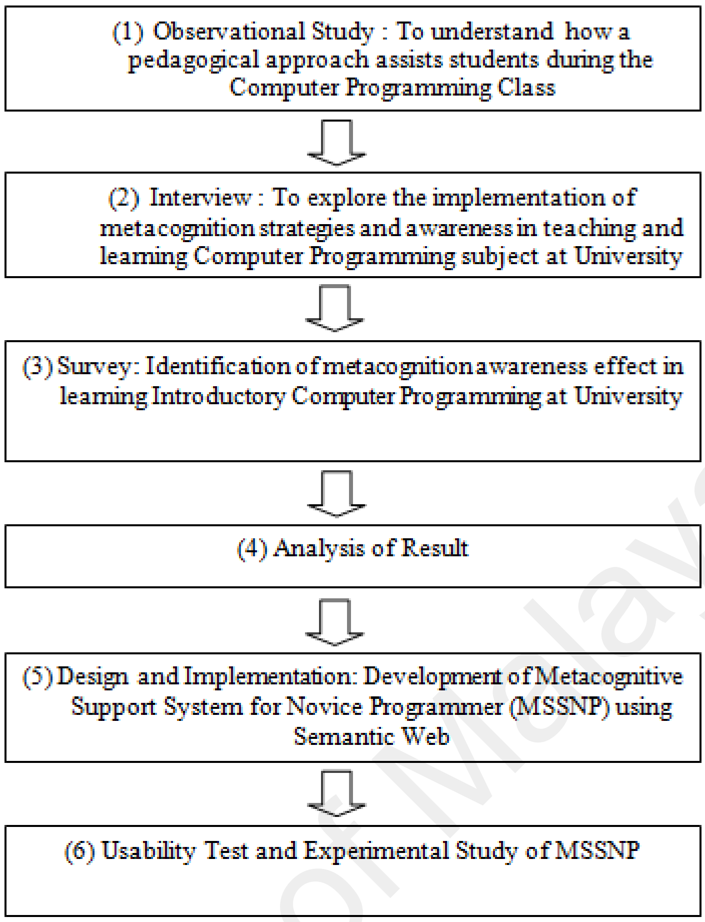

(1) Observational Study : To understand how a pedagogical approach assists students during the Computer Programming Class
(2) Interview : To explore the implementation of metacognition strategies and awareness in teaching and learning Computer Programming subject at University
(3) Survey: Identification of metacognition awareness effect in learning Introductory Computer Programming at University
(4) Analysis of Result
(5) Design and Implementation: Development of Metacognitive Support System for Novice Programmer (MSSNP) using Semantic Web
(6) Usability Test and Experimental Study of MSSNP

Figure 3.1: Research framework for Metacognitive Support Environment for Novice Programmers

Five steps have been identified in order to achieve the objectives of this research work (as illustrated in Figure 3.1). The observational study on how a pedagogical approach assists students during the computer programming class is further discussed in the following section.

3.1 Observational Study

The main objective of this study is to obtain further understanding concerning how a pedagogical approach assists students during the computer programming class. The observational study was conducted in the classroom during the Introductory Computer Programming class where the instructor was informed in advance. This study was conducted at the Faculty of Computer and Mathematical Sciences, UiTM, Shah Alam.

A letter requesting consent (See Appendix A) to conduct the survey was given to the instructor and positive feedback was received from the instructor by allowing us to do the observation study during the class sessions. The teaching and learning activities during the class session were observed and recorded. The main objective of this study was partly derived from the motivating factor that novice programmers' would want to have a supporting environment in learning computer programming (further discussed in Chapter 4). The respondents participating in this study were students who had been taught by the instructor. The students were first year undergraduates who took Introductory Computer Programming. The questions to be answered in this study are:

1. How do students learn during the computer programming class?
2. What are the positive attributes derived from each learning approach?
3. What are the negative attributes derived from each learning approach?

The observational study was conducted for five different class sessions. The results of the observation are discussed further in the following section.

3.1.1 Results of the Observation

The results of the study are analyzed and summarized in Table 3.1. The analysis of the onservation is done quantitatively based on the notes jotted down during the survey. The findings of the survey show that three types of learning approach are typically and commonly used by instructors to teach computer programming; cooperative learning, discussion and dialog that takes place during the lecture and repetition. Collaboration learning has been proven to be effective for all types of students because it promotes learning and fosters respect and friendship among diverse groups of students. Several studies show that cooperative learning encourages metacognitive thought (Veenman et al., 2006).

Table 3.1: Pedagogical approach in teaching and learning Computer programming at university

How did students learn during class?	Positive attributes of the learning	Negative attributes of the learning
Collaboration	• Working with others that were on the same level • Sharing ideas • Assistance from other students • Learning from others • Promote metacognitive reflecting	• Needed to spend more time collaborating with others • People needed to go at a different pace and speed
Discussion and dialog	• Students feeling comfortable voicing their opinions • Build self-confidence • More interactive • Promote metacognitive reflecting	• Passive students who are too shy to answer questions and raise an opinion get lost in large-group context
Repetition	• Repeating lecture builds metacognitive skill as it allows the student an opportunity to assess what they understood when the lecture was first received as opposed to the repeated version	• Too much depends on lecturer/instructor

However, with this learning approach, more time is needed for collaborative learning with others in that some students need to go at a different speed while working in the group. Someone is either slowed down or forced to catch up faster than they would like to. The second learning approach is through discussion and dialogue. The benefits of this approach are that the student feels comfortable voicing this, which, indirectly, could help students to build their self-confidence, promote metacognitive learning, and, with this approach, the learning atmosphere is more interactive. However, the drawback of this approach is that unresponsive and passive students who are too shy to get involved and to answer questions or raise an opinion are often lost in the large-group context and usually do not receive the necessary attention compared to extrovert students. The third approach is through repetition. Repeating lectures build metacognitive skills as it allows the student an opportunity to assess what they understood when the lecture was first received as opposed to the repeated version. However, this type of learning approach requires more time for repeating the lecture and discourages students from becoming independent. Literature that solely focuses on novice programmers learning programming metacognitively is lacking. Therefore, based on the above findings, a study on a metacognitive support environment in learning computer programming is needed for novice programmers in learning programming, as classroom learning alone is not sufficient. Thus, a study is perceived to be necessary, specifically in the following areas:

1. Establishing the computer-based metacognitive activities in supporting computer programming learning.
2. Identification of methods in learning programming metacognitively
3. Identification of the motivating factors in using a support system for the purpose of learning programming

3.2 Interview

This section is divided into two sections. The first section discusses the method of conducting interview sessions with the expert lecturers and the second section discusses the approach taken for interviewing the novice programmers.

3.2.1 Interviews with Expert Lecturers

The main objective of the interview with expert lecturers is to explore the metacognitive implementation and metacognition awareness in teaching computer programming courses at universities. The interviews took place in the FSKM, UiTM. The participation was voluntary in nature and each session lasted around 45 minutes. The interview sessions were conducted during the midterm of semester. Each interview session was conducted individually. Five expert lectures were chosen for the interviewing sessions. The process of choosing the participants is based on the years of experience in teaching computer programming courses and the involvement of research projects and consultancy. Three of them are PhD holders and the others are master degree holders. All participants have teaching experience of more than five years in various types of programming languages, such as structured programming, JAVA programming, C, and C++ programming and assembly languages. The questions asked during the interview sessions are:

1. Have you ever heard of metacognition?
2. Would you describe what metacognition is? (After reading the definition)
3. How is metacognition important for a lecturer in teaching?
4. Do you feel that metacognition is important in computer science education? Why?
5. How do you teach metacognitive skills to your students to improve their learning?
6. How do you apply metacognition in your own teaching?
7. When you teach, what is important to you?
8. What are you thinking about when you are teaching?
9. Before you teach, what do you usually do?
10. What do you think about before you teach?
11. After you teach, what do you usually do?

3.2.2 Interviews with Novice Programmers

The objective of these interviews is to understand the learning behavior of the novice programmers in learning computer programming. The questions in this study were extracted from the motivating factors that novice programmers' would want to have as supporting features, which is discussed in more detail in Chapter 4. Fifteen emails were randomly sent out to participants for the purpose of inviting them to join the observational study. Only five participants agreed to be part of the study while the others refused. Five questions were formulated and given as experimental questions to the respondents. The questions were designed based on three types of metacognition knowledge, as categorized by Flavell (1979):

1. Person variable, which refers to what one recognizes about his or her strengths and weaknesses in learning and processing information.
2. Task variable, which refers to what one knows or can figure out about the nature of a task and the processing demands required to complete the task.
3. Strategy variable, which refers to the strategies a person has “at the ready” to apply in a flexible way to successfully accomplish a task.

Livingston (1997) provides an example of all three variables: “I know that I (person variable) have difficulty with word problems (task variable), so I will answer the computational problems first and save the word problems for last (strategy variable).” The questions are also obtained from the analysis of the interview with the expert lecturers.

The questions to answer are:

1. Do you know your own strengths and weakness in learning programming?
2. How do you motivate yourself to learn programming?
3. What are the types of resource that you usually refer to?
4. What techniques do you use to process information in learning programming?
5. What kind of help seeking method do you use to make you understand while learning programming?
6. Do you track the amount of time taken for solving problems in programming?
7. How do you evaluate your own knowledge performance and understanding?

The respondents who participated in the survey were chosen from the undergraduate students that took the programming subject (further described in Chapter 4).

3.3 A Questionnaire Survey using MAI Inventory

This research aims to explore the effect of metacognitive awareness on the learning success of computer programming education at universities. Effect in this study referred to the result or the consequence of metacognitive awareness of an individual in learning computer programming. It is measured by identifying the relationship between total score of metacognitive awareness level of students with their GPA. The question that was highlighted and put forward in this research work is 'Do metacognitive skills have a positive effect on students' learning success in introductory computer programming at university?' The best practice suggested by (Gaddis, 1998) in designing the questionnaire survey was applied as follows: 1) Introduction of the study is provided at the main page of the survey form to encourage participation from the respondents; 2) using filtering questions to direct respondents to appropriate questions; 3) dividing long surveys into sections; 4) using appropriate open-ended questions; 5) ending the questionnaire by thanking the respondents for their time; 6) giving a token of appreciation to encourage more response; and, finally, 7) pre-testing the questions before the actual survey.

The questionnaire survey is divided into two sections. The first section question concerns the demographic information of the respondents and the second section consists of the self-report questions that are taken from the Metacognition Awareness Inventory (MAI) and augmented with some additional information requests such as demographic information, types of strategies used for learning Computer Programming and motivational factor of using support tool. The inventory was invented by (Schraw &

Sperling Dennison, 1994). The self-questionnaire in MAI is well structured, convenient, easy to use and secure. It has a well-devised statement and well-validated inventory (Hughes, 2015; Stewart & Hadley, 2014; Young & Fry, 2012) . The MAI is also a general domain inventory for assessing the knowledge of cognition and regulation of cognition of individuals. It consists of a 52-item self-report tapping into two components of metacognition; the metacognitive knowledge and metacognitive regulation. Their study reveals that the knowledge cognition factor and the regulation of cognition factor have strong support for each other. The questionnaire survey includes items regarding knowledge and regulation of cognition and is divided into eight component processes (Schraw & Sperling Dennison, 1994).

Knowledge about cognition is composed of three subcomponent processes that facilitate the reflective aspect of metacognition: declarative knowledge (i.e. knowledge about self and strategies), procedural knowledge (i.e. knowledge about how to use strategies'), conditional knowledge (i.e. knowledge about when and what strategies). Whereas the regulation about cognition component is the aspect of learning including planning, information management strategies, comprehension monitoring, debugging strategies and evaluation (Schraw & Sperling Dennison, 1994). The decomposition of the MAI is illustrated in Figure 3.2. Undergraduate students of Computer Science from several universities were invited to participate in this study. Questionnaire surveys (Refer Appendix B) were distributed accordingly.

The sampling of the respondents was based on a convenience sample as it requires less effort and time, which was considered important in this study. Self-selected or convenience sampling is based on the voluntary act of respondents to the uncontrolled distribution of the instrument, i.e. a questionnaire (Schonlau, Ronald Jr, & Elliott, 2002). The pilot test in this study is relatively important to check the effectiveness of the MAI instrument to avoid misunderstanding of the questions, to check the clarity of the question and to test how long it takes to be completed. The first thing to do was to mock-up the online survey by sending out through the email to a small group of the targeted respondents, to ensure that there are no obvious problems with the questions. Once this had been done, a test was carried out on larger group of people.

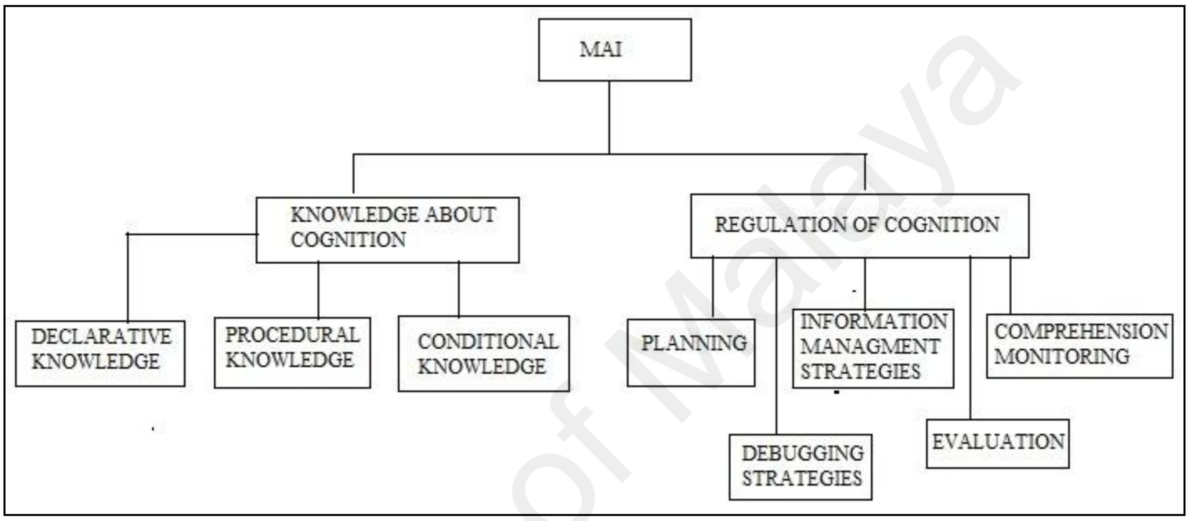


Figure 3.2: The decomposition of Metacognitive Sub Components (Tobias S. et al., 1999)

Researchers investigate metacognition knowledge and awareness and how it relates to measure academic success. In these studies, the skills of metacognition are measured from the perspective of metacognitive regulation and metacognitive knowledge. Given the discovery thus far regarding academic achievement measured with MAI, the primary goal of this study was exploratory in nature. This study is interested in determining the correlation between the MAI and the Grade Point Average (GPA) of the introductory Computer Programming course, understanding the learning behavior of students regarding metacognition in learning computer programming as well as identify the motivation that would influence the novice programmer to use a support system during computer programming learning.

The tabulation of the question from the survey is provided in Table 3.2. The first section is about the demographic study, in this section the general background on respondents' academic year of study for undergraduates' student. (Info: Gender, GPA, University, Year of Semester) are captured. The second section of the questionnaire survey is about Cognitive Knowledge and Cognitive Regulation of an individual, during this part, respondents are exposed to eight components of metacognitve awareness as described by (Tobias S. et al., 1999). In the last section, respondents are required to provide their strategies in learning Computer Programming as well as the motivational factor of using support tool.

Table 3.2: Tabulation of Questions from the Survey Questionnaire

Section	**Description**	**Type**	**Literature**
Demographic Study	The general background on respondents' academic year of study for undergraduates student. (Info: Gender, GPA, University, Year of Semester)	Demographic Study	(Soloway & Spohrer, 2013)
Cognitive Knowledge	Knowledge about oneself as a learner and factors affecting cognition	Declarative Knowledge	(Dechant, 2013; Schraw, 2006; Schraw & Moshman, 1995; Slavin & Davis, 2006)
	Awareness and management of cognition, including knowledge about strategies	Procedural Knowledge	(Banks & Millward, 2007; Dechant, 2013; Kuhn & Dean, 2004; Schraw, 2006)
	Knowledge about why and when to use a given strategy	Conditional knowledge	(Dechant, 2013)

Cognitive Regulation	Identification and selection of appropriate strategies and allocation of resources	Planning	(Cross & Paris, 1988; Paris & Winograd, 1990; Schraw, 2006)
	Attending to and being aware of comprehension and task performance	Comprehension Monitoring	(Cross & Paris, 1988; Paris & Winograd, 1990; Schraw et al., 2006; Schraw & Moshman, 1995; Whitebread et al., 2009)
	The skills and strategy sequences used to process information more efficiently (e.g. organizing, elaborating, summarizing, selective focusing)	Information Management Strategies	(Schraw, 2006)
	Strategies used to correct comprehension and performance errors	Debugging Strategies	(Schraw., 2006) (Schraw & Moshman, 1995)
	Assessing the processes and products of one's learning, and revisiting and revising learning goals	Evaluation	(Cross & Paris, 1988; Paris & Winograd, 1990; Schraw et al., 2006; Schraw & Moshman, 1995; Whitebread et al., 2009)
Types of strategy in learning programming	Typical strategies used to learn computer programming		(Robins et al., 2003)
Motivating factor	Motivating Factor in using support system for learning computer programming		(Ramaha & Ismail, 2012)

Spearman's Rho, which is a non-parametric correlation analysis, was conducted to prove that a relationship exists between the knowledge of cognition and the regulation of cognitive factors, correlation between the MAI and Grade Point Average (GPA) for Introductory of Computer Programming Course, as well as the relationship between the subcomponents. The sampling is based on convenience sampling, that is, a non-probability sampling that consumes less time and effort, which is considered to be of great significance in this study. This sampling method is based on the respondents voluntary act to the instrument distribution that is uncontrollable, i.e. a questionnaire (Schonlau et al., 2002).

The online survey is set up using the Google Docs' application. The Google doc's application allows researchers to design an online survey form and invite a group of people to respond via email or the URL that is created can be posted on Facebook. The targeted respondents were undergraduate students of Computer Science from fourteen selected universities, and the responses were assembled automatically into a Google Docs' spreadsheet. The participants were invited through email and a group on Facebook. With the permission given by the academic affairs departments of the universities, the email addresses of students were obtained from the student database system. The main objective of the experimental study is to identify the metacognitive effect towards the learning success of the Introduction of Programming subject at university using the Metacognitive Awareness Inventory (MAI) developed by (Schraw & Sperling Dennison, 1994).

3.3.1 Survey Methodology

This section discusses the methods used for data collection for the questionnaire survey. The objective of the survey is to identify the metacognitive effect towards the learning success of computer programming. The questionnaire is further divided into nine sections. The first section concerns the demographics. This study was conducted using a descriptive research model. The sampling was based on convenience sampling, that is, non-probability sampling, as it consumed less time and effort, which were considered of great significance in this study. This sampling method is based on the respondents voluntary act to the instrument distribution that is uncontrollable, i.e. a questionnaire (Schonlau et al., 2002). The online survey was set up using the Google Docs' application. The Google Docs' application allows researchers to design an online survey form and invite a group of people to respond via email or the URL created can be posted to Facebook. All responses are assembled automatically into a Google Docs' spreadsheet. The participations were invited through email and groups on Facebook. With the permission given by the academic affairs departments of the universities, the email addresses of students were obtained from the student database system.

The questionnaire comprises two sections. The first section is to elicit personal information about the respondents including educational background. The second section consists of 52 self-reports in which students are required to rate as TRUE or FALSE. The MAI is given 1 (one) point for each TRUE on the chart and for each question with a FALSE answer, a 0 (zero) score is given. The second of the questionnaire is regarded to the two components of knowledge discussed above; knowledge about cognition and regulation of cognition. The scores are calculated by adding the total scores from each factor. Higher total scores of each factor correspond to greater metacognitive knowledge and greater metacognitive regulation. Table 4.1 presents the subcomponents of MAI.

Table 3.3: Subcomponents of MAI

Instrument Metacognitive Inventories Subscales	**No of Items**
Procedural Knowledge	4
Declarative Knowledge	8
Conditional knowledge	5
Planning	7
Comprehension Monitoring	7
Evaluation	6
Debugging Strategies	5
Information Management Strategies	10
Total	52

3.3.2 Procedure

The MAI survey was set up using the Google Docs' application for respondents to access. The Google Docs' application allows researchers to design an online survey form and invite a group of people to respond via email, or the URL created can be posted to Facebook. All responses are assembled automatically into a Google Docs' spreadsheet. The URL of the online survey was sent through the student group and individual email. The URL was also posted on the Faculty of Computer Science Facebook page that come from different groups/universities. The advantages of the Web-based survey method are as follows:

- Convenience: This technique represents a convenient and efficient way of reaching potential respondents. They are able to receive the questionnaire and complete it in the privacy of their home (Rea & Parker, 2012)
- Rapid data collection: information, especially information that must be timely can be collected and processed within days (Rea & Parker, 2012)

- Cost-effectiveness: This technique is more cost-effective than the traditional mail survey because there is no need for postage or paper supplies (Rea & Parker, 2012)
- Ample time: The respondent is not pressed for time in responding to the Web-based survey and has the opportunity to consult records in answering the questions. There is time to consider response choices and to respond to open-ended questions in the form of text (Rea & Parker, 2012)
- Confidentiality and security: Potential respondents can be reminded to respond so the respondents can be protected on a secure server through the efforts of the research team (Rea & Parker, 2012)
- Specialized populations: The Web-based survey is particularly useful in reaching specialized or well-identified populations whose e-mail addresses are readily available (Rea & Parker, 2012)
- Complexity and visual aids: Web-based surveys can utilize visual images and more complex questions (Rea & Parker, 2012)

3.4 Design and Implementation of Metacognitive Support System for Novice Programmer (MSSNP)

The design of the suggested system incorporated the metacognitive elements and activities that would assist novice programmers in learning programming metacognitively. The proposal for an ideal architectural design is illustrated in figure 3.3, which is refined from the findings in Chapter 4. The design and implementation of the Metacognitive Support System for Novice Programmer (MSSNP) is further discussed in Chapter 5.

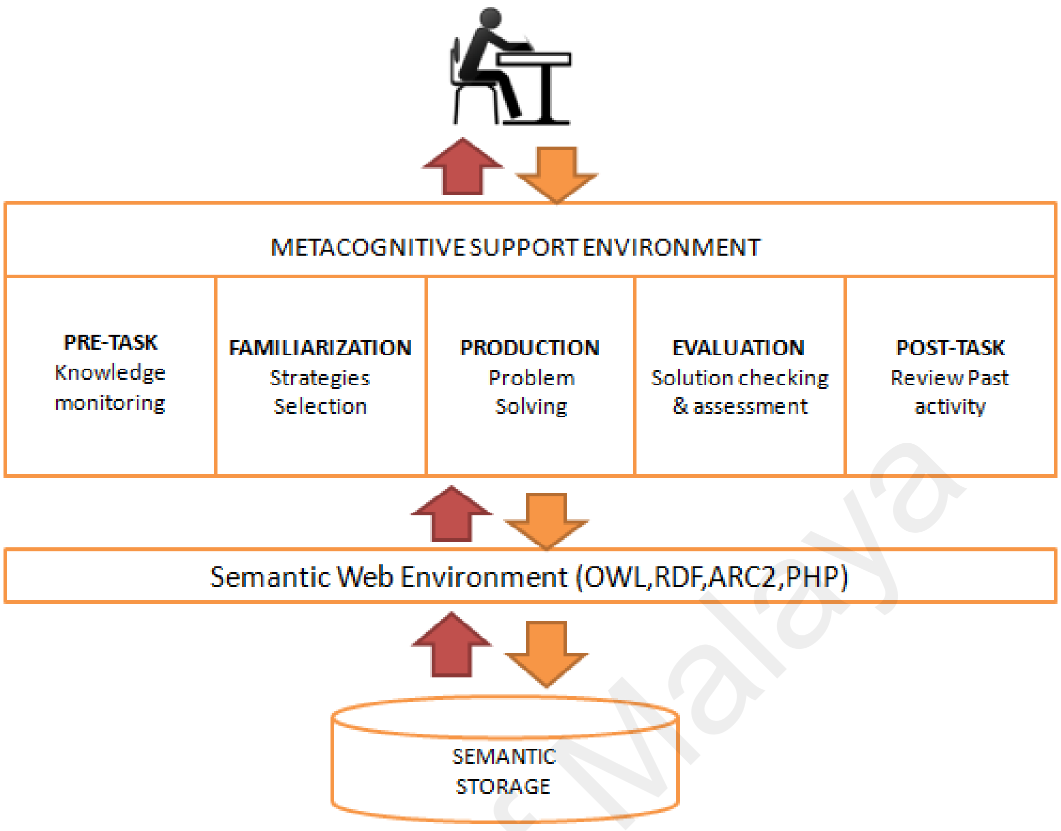

METACOGNITIVE SUPPORT ENVIRONMENT
PRE-TASK Knowledge monitoring
FAMILIARIZATION Strategies Selection
PRODUCTION Problem Solving
EVALUATION Solution checking & assessment
POST-TASK Review Past activity
Semantic Web Environment (OWL,RDF,ARC2,PHP)
SEMANTIC STORAGE

Figure 3.3: Proposed architectural design of Metacognitive Support System for Novice Programmer (MSSNP)

3.5 Experimental study and evaluation of the MSSNP

The previous chapters have discussed the characteristics of designing a metacognitive instruction for learning programming. We develop the MSSNP following a rationale that metacognitive activities encourage students to think about their knowledge monitoring ability, to select metacognitive strategies and to evaluate their learning experience that have a positive impact on their learning gain. Because it is difficult to test changes in the state of metacognition, an experimental study was established to test whether the design model was effective. The empirical evaluation of students' interaction with the MSSNP focused mainly on the observation of metacognitive and performance changes. For the analysis of the results, several inferential statistical measurements, such as Kolmogorov Smirnov (data normality test), Shapiro-Wilk (data normality test), Spearman's rho, and Wilcoxon were employed. The experiment was undertaken with undergraduate students from the Faculty of Computer and Mathematical Science, UiTM, Shah Alam. The design of the experiment is presented in Chapter 6. It also presents the statistical analysis. The observation and analysis are used to formulate more general answers for the research questions. Apart from the experimental test, the users perception of the system usability is important in this study to ensure the user's acceptance of the system. Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users. It was defined by ISO 9241 as "*the extent to which a product can be used by specified users to achieve goals with effectiveness, efficiency and satisfaction in the specified context of use*" (Veenendaal, 1998). The Software Usability Measurement Inventory (SUMI) is a comprehensive solution for measuring the quality of the software from the end user's perspective. SUMI is an internationally proven method and well tested instrument for determining the software quality from the end user. As already mentioned in the previous section, the SUMI questionnaire includes 50 items to represent the efficiency, affects, helpfulness and learnability attributes of the software. The statements presented to the users are about their views and behavior of the tested software.

3.6 Summary

As discussed in previous chapters, the objective of this study is to identify methods of supporting novice programmers learning computer programming metacognitively. The six steps involved in this study are:

1. A questionnaire survey is conducted to identify the metacognitive effect on the learning success of computer programming, to identify the novice programmer learning behavior, to identify the motivating factor of utilizing the support system in learning computer programming.

2. Interviews with expert lecturers are conducted with the aim to investigate the importance of metacognition in teaching and learning, the awareness and the application of metacognition in teaching and learning programming
3. Analysis of the results and interviews.
4. Development of prototype system that provide features derived from the requirement analysis study.
5. Evaluation of the system in terms of metacognitive changes through the interaction with the prototype system and usability measurement of the prototype system.

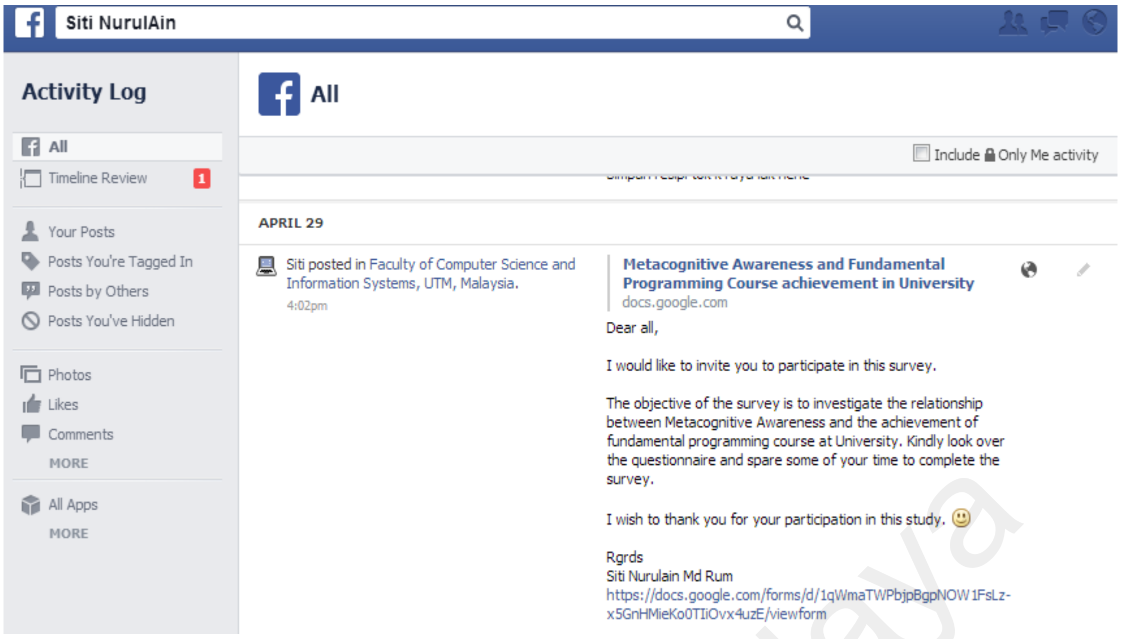
Support features that would assist novice programmers in learning computer programming that were elicited from the survey results give vital information in this research work. In addition, the learning behavior of novice programmers provides an insight for refining the support features by incorporating the learning elements that relate to metacognition. The details of the results and analysis of the survey are discussed further in the following chapter (Chapter four).

CHAPTER 4: ANALYSIS OF METACOGNITIVE AWARENESS IN TEACHING AND LEARNING COMPUTER PROGRAMMING

(Garner & Alexander, 1989) emphasize the relevance of empirical research on the measurement of metacognition, and suggested that the following questions should be addressed: How to accurately measure "knowledge about knowledge"? How can the effectiveness of strategies used in training be measured? This chapter discusses the empirical study conducted to explore the eight components of knowledge of novice programmers that comprise declarative knowledge, procedural knowledge, conditional knowledge, planning, information management strategies, comprehension monitoring, debugging strategies and evaluation. As discussed in Chapter 3, the Metacognitive Awareness Inventory (MAI) (Schraw & Sperling Dennison, 1994) is used in this study measuring novice programmers' awareness of metacognition in learning the introductory computer programming at the university. The MAI inventory consists of 52-items of self-report that tap into two components of metacognition: the metacognitive knowledge and metacognitive regulation (See Appendix B).

4.2 Results and analysis of questionnaire survey

In Chapter 1, we defined the characteristics of novice programmers as someone that lacks the knowledge and skills of programming knowledge. Thus, the target population in this study is the undergraduate students of computer science that already took the introductory computer programming at university or education institution. Realistically, to take the entire population is too large for a researcher to attempt to survey all members; therefore, a chosen sample is used to represent the population. In this study, 400 (sample size) sets of questionnaires were distributed to the undergraduate students of computer science via online. Using an Internet survey tool, we were able to place a date and time stamp on each response and find out how much time it took each person to complete it. We know from past experience that respondents who complete the survey too quickly (less than 30%-50% of median time) are likely to not be reading or answering the questions appropriately. The same is true for flat-liners (i.e. those who mark each answer the same), which are often speeders. They may have read the questions, but they do not really think about their answers. Therefore, it was considered prudent to remove speeders and flat-liners from our data to eliminate a lot of meaningless data. After the data cleansing process, which involves the activity of removing ‘duplicate responses’, ‘speeders’ and ‘flat-liners 164 of the respondents were chosen for further analysis. According to (Visser, Krosnick, Marquette, & Curtin, 1996) who showed that surveys with lower response rates (near 20%) yielded more accurate measurements than did surveys with higher response rates (near 60 or 70%) and surveys with low response rates are not necessarily low in validity. The respondents were invited through Facebook (see Figure 4.1) and the invitation was posted to the computer science Facebook groups (see Figure 4.2 as an example).



Siti posted in Faculty of Computer Science and Information Systems, UTM, Malaysia.
4:02pm

Metacognitive Awareness and Fundamental Programming Course achievement in University
docs.google.com

Dear all,

I would like to invite you to participate in this survey.

The objective of the survey is to investigate the relationship between Metacognitive Awareness and the achievement of fundamental programming course at University. Kindly look over the questionnaire and spare some of your time to complete the survey.

I wish to thank you for your participation in this study.

Rgrds
Siti Nurulain Md Rum
https://docs.google.com/forms/d/1qWmaTWPbjpBgpNOW1FsLz-x5GnHMieKo0TIiOvx4uzE/viewform

Figure 4.1: Survey Invitation via Facebook

This page is open to all Student of Bachelor of Computer Science [Kuliyyah of Information and Communication Technology IIUM]

Figure 4.2: The IIUM Bachelor of Computer Science Facebook group

The distribution of the chosen respondents by University is presented in Table 4.1,

Table 4.1: Distribution of respondents group by University

Universities	Frequency	Percentage (%)
International Islamic University Malaysia (IIUM)	7	4.27
Kolej PolyTech MARA Kuala Lumpur	5	3.05
UiTM Kedah Campus	3	1.83
UiTM Kelantan Campus	6	3.66
UiTM Perlis Campus	8	4.88
UiTM Sarawak Campus	18	10.98
UiTM Segamat Campus	34	20.73
UiTM Sri Iskandar Campus	20	12.20
UiTM Terengganu Campus	16	9.76
UiTM Pahang Campus	15	9.15
UiTM Melaka Campus	16	9.76

Univerisity of Malaya (UM)	6	3.66
Universiti Teknologi Malaysia (UTM)	4	2.44
Universiti Sains Malaysia (USM)	6	3.66
Total	164	100

The majority of the respondents come from the UiTM Segamat Campus, which make up 12.2%, followed by the respondents from UiTM Sri Iskandar Campus (12.2.%). In this study, we employed an explanatory study to explain and measure the relationships between variables using statistical techniques. Two types of statistic – descriptive and inferential statistics – were used to analyze the results. The descriptive statistics were carried out to visualize what the data showed and to see the pattern that might emerge from the data, while the inferential statistics were used to analyze the results obtained and draw the conclusion. We used SPSS (Version 10) to perform the descriptive analysis and explanatory analysis. The findings are reported and discussed accordingly in the following section in more detail.

4.2.1 Demographic Study

Appendix C provides the demographic background of the respondents who participated in the survey. This study was participated in by 164 undergraduate students of computer science (102 females, 62 males), as shown in Table 4.2. The target sample of this study is 400 and the response rate was 41%. For an online survey, conventionally, a 40% response rate is considered as a good response rate (Deutskens, De Ruyter, Wetzels, & Oosterveld, 2004). As depicted in Table 4.1, the majority of respondents (82.93%) are from Universiti Teknologi Mara from nine different campuses (Segamat Campus, Kelantan Campus, Perlis Campus, Sarawak Campus, Terengganu Campus, Pahang Campus and Melaka Campus), 4.27% are from International Islamic University Malaysia, 3.05% from Kolej PolyTech MARA Kuala Lumpur, 3.66% from University of Malaya and 2.44% from Universiti Teknologi Malaysia. Table 4.2

presents the distribution of respondents by the University's Introductory Computer Programming course. The distribution of respondents grouped by the GPA is presented in Table 4.6: 26.83% are from GPA between 2.00 – 2.49, 34.15% students in the 2.5 – 2.99 GPA range, and 24.39% student in the 3.5 – 4.00 GPA range. The distribution of respondents by gender, GPA and academic year of study is presented in Appendix C.

Table 4.2: Distribution by University's programming course

Course Code	**Course Title**	**Course description**	**University**	**Frequency**	**(%)**
CSC 1101	Structured Programming Language/ Bachelor of Computer Science (BCS)	This course focuses on the fundamentals of structured programming with C++. Students are taught the art of problem-solving in programming, the techniques, architectures, the design issues and fundamentals about class and object. This course is designed to prepare a student to be familiar with software development process.	Internationa l Islamic University Malaysia	7	4.27
SC128	Fundamentals of Computer Problem-solving / Diploma of Computer Science (CS110)	This is fundamental to problem-solving courses using computers via structured programming. The focus is on various aspects of problem-solving rather than syntactical aspect of the chain programming language, mainly consisting of the	UiTM Kedah, UiTM of Kelantan, UiTM of Perlis, UiTM of Sarawak, UiTM of Segamat Johor, UiTM of Sri Iskandar Perak, UiTM of	136	82.93

		problem domain, phases of problem-solving and basic techniques in designing solutions.	Terengganu, UiTM of Pahang, UiTM of Melaka, Kolej PolyTech MARA Kuala Lumpur		
WXES1116	Programming I/ Bachelor of Computer Science (AI) (MC00)	This course is a basic to Object Oriented Programming using JAVA. It defines the concepts of Object Oriented programming with flowchart and pseudocode. The student learns how to write and develop programs using the appropriate semantic and syntactic features.	University of Malaya (UM)	6	3.66
SCJ1013	Programming Techniques I/ Bachelor of Computer Science (SCJ)	This course equips the students with theory and practice on problem-solving techniques by using the structured approach. Students are required to develop programs using C++ programming language	Universiti Teknologi Malaysia (UTM)	4	2.44
CPT111/3	Principles of Programming/ Bachelor of Computer Science	In this course, the student is equipped with the principle of Programming. They are taught how to analyze and examine problems, and transform the problem solution into codes that are understood by programming	Universiti Sains Malaysia (USM)	6	3.66

		language			
Total				164	100

4.2.2 Data Normality Test

The inferential statistical measurement (Kolmogorov Smirnov and Shapiro-Wilk) is performed to test the normality of GPA and each other component of MAI. The results in Table 4.3 show that most of the variables were not normally distributed; therefore the non-parametric type of statistic called Spearman Rho is applied for further analysis.

Table 4.3: Tests of Normality

Attr.	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
GPA	.214	164	.000	.866	164	.000
P	.221	164	.000	.838	164	.000
MAI	.106	164	.000	.922	164	.000
DL	.138	164	.000	.911	164	.000
CDL	.231	164	.000	.841	164	.000
PL	.202	164	.000	.874	164	.000
IMS	.178	164	.000	.903	164	.000
CM	.172	164	.000	.858	164	.000
DBG	.281	164	.000	.725	164	.000
EVL	.181	164	.000	.903	164	.000
a. Lilliefors Significance Correction						

4.2.3 Descriptive statistics

Of the 164 respondents, the mean MAI score was 36.51, the mean score for the knowledge of cognition factor (KC) was 11.76 and regulation of cognition factor (RC) was 24.75, Procedural (P) was 3.02, Declarative (DL) was 4.94, Conditional (CDL) was 3.76, Planning (PL) was 4.67, Information Management Strategies (IMS) was 7.38, Comprehension Monitoring (CM) was 4.74 and Debugging Strategies (DBG) was 3.93, respectively. Table 4.4 presents the descriptive statistics of all the other variables. The value of mode, standard deviation, skewness and coefficient variation for all variables is presented in Table 4.4. In the theory of statistics, skewness is a measure of the asymmetry of the distribution; it can be a positive or negative value. The value of skewness can be used to define the extent to which a distribution differs from a normal distribution. The values in Table 4.4 are translated into graphs, which are presented in Figure 4.3 and Figure 4.4. The graphs that are symmetrically bell-shaped can be classified as normal distribution where the mean and mode are equal.

Table 4.4: Means, Mode, Standard Deviations, Skewness and coefficient variations of all variables

Variables	**Mean**	**Mode**	**Std Dev**	**Skewness**	**Coefficient variation**
GPA	2.88	2.745	0.50916	0.299	17.68%
MAI	36.51	48	10.660	-0.568	29.20%
Knowledge about Cognition (KC)	11.76	17	3.507	-0.150	29.82%
Regulation of Cognition (RC)	24.75	10	7.592	-0.762	30.67%
Procedural (P)	3.02	3	0.913	-0.587	30.23%
Declarative (DL)	4.94	6	2.033	-0.027	41.15%
Conditional (CDL)	3.76	5	1.224	-0.619	32.55%
Planning (PL)	4.67	6	2.025	-0.588	43.36%
Information Management Strategies (IMS)	7.38	9	1.973	-0.296	26.73%
Comprehension Monitoring (CM)	4.74	7	2.047	-0.660	43.18%
Debugging Strategies (DBG)	3.93	5	1.389	-1.243	35.34%

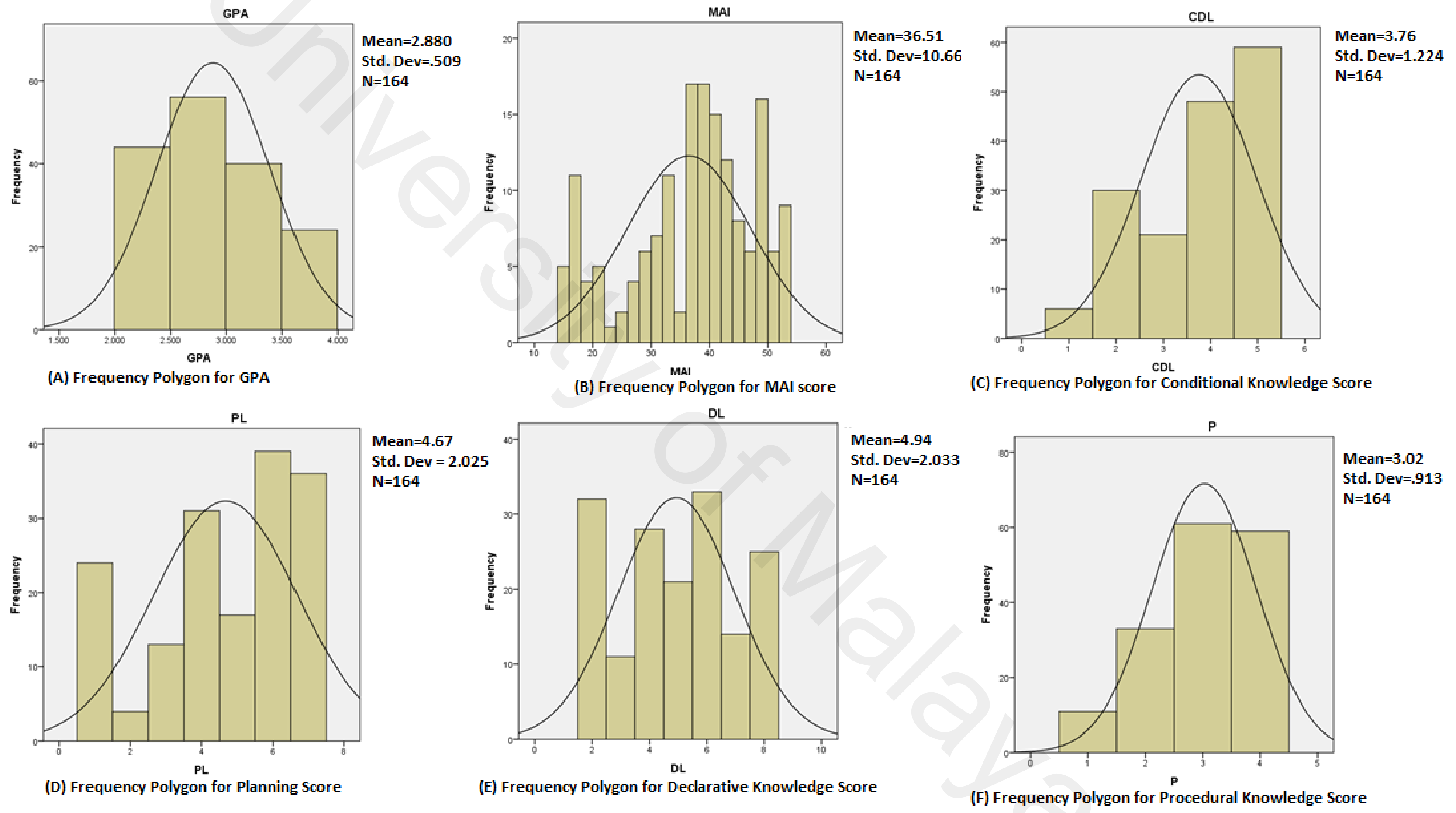
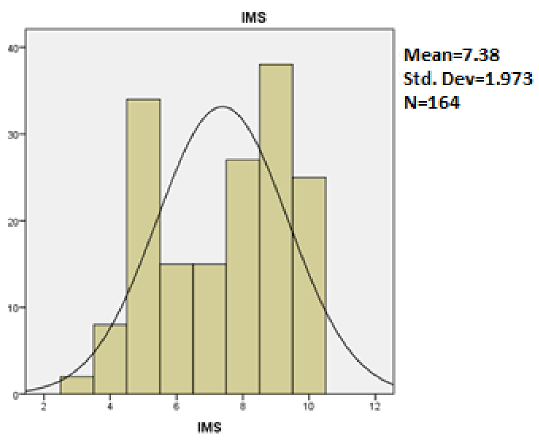
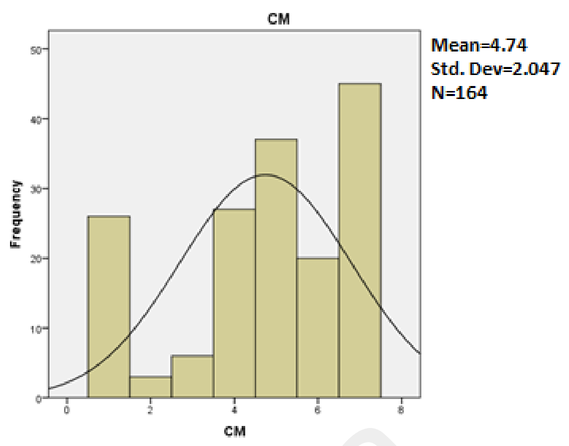


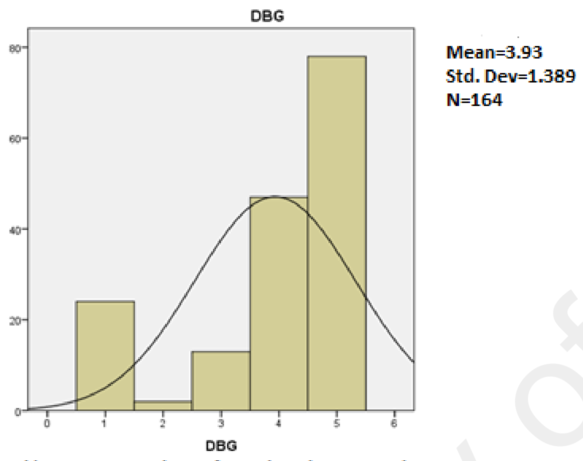
Figure 4.3: Frequency Polygon for GPA, MAI, CDL, PL, DL and P score



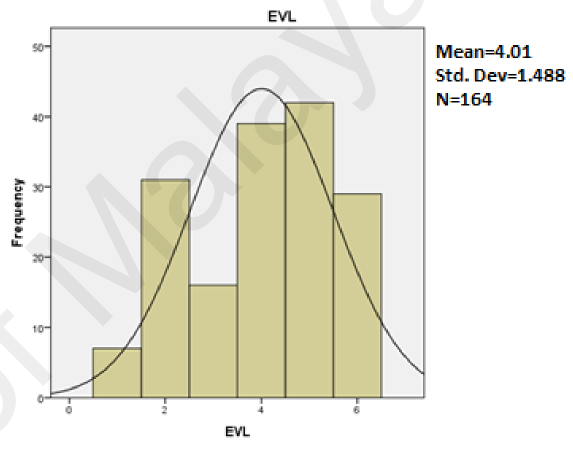
(G) Frequency Polygon for Information Management Strategies



(H) Frequency Polygon for Comprehension Monitoring Score



(I) Frequency Polygon for Debugging Strategies score



(J) Frequency Polygon for Evaluation Score

Figure 4.5: Frequency Polygon for IMS, CM, DBG and EVL score

Figure 4.3 and Figure 4.4 show the data distribution frequency for the MAI score and other individual MAI sub-component scores. The frequency polygon diagrams were provided in this thesis for understanding the shapes of distribution for each variable. The black line superimposed on each of the histograms in Figure 1 and Figure 2 represent the bell-shaped "normal" curve. All diagrams (as shown in Figure 1 and Figure 2) depicted that each variable (GPA, MAI, P, DL, CDL, PL, IMS, CM, DBG and EVL) is normally distributed, where the mean and median are almost the same. In the theory of statistics, the coefficient of variation (CV) is a standardized measure of dispersal of a probability distribution, or, in other words, it can be defined as the ratio of the standard deviation to the mean. In examining the coefficient variation (CV) result, most of the CVs are approximately at 30%, which implies that the data are stable and that the standard deviation compared to the mean is acceptable. Based on the normality test the non-parametric statistic is the most suitable statistic used for further investigation. This type of statistic does not require the distribution of population to be characterized by certain assumptions, for instance, they do not assume that the outcome is approximately normally distributed compared to parametric tests that require a specific probability distribution, such as 'normal distribution'. The Spearman's Rho, which is a non-parametric correlation analysis, was conducted to prove that the relationship existed between the knowledge of cognition and the regulation of cognitive factors, correlation between the MAI and Grade Point for Introductory of Computer Programming Course (GPA) as well as the relationship between the subcomponents.

4.2.4 Correlation Coefficient Result

The correlation coefficient results for all variables is presented in Appendix D. In Table 4.5, the results indicate that there is a positive linear correlation between the GPA and MAI score with a correlation coefficient of r= 0.8226 and significant at the 1% level. The findings confirmed that metacognition has a positive effect on students learning success in the Introductory Computer Programming course at university, as the MAI scores go up, the GPA tends to increase as well and vice versa. The variations in MAI explained at 67.67% of the variation in GPA (r^2 = 0.6767 with n=164), indicating that there is a possibility of 32% of other factors influencing or affecting student learning success in the Introductory to Computer Programming at University. The results also tell us that a strong correlation exists between the GPA and Knowledge about Cognition (KC) with r = 0.7483, GPA with Regulation of Cognition (RC) with r = 0.8224, GPA and Procedural Knowledge (P) with r = 0.4387, GPA and Declarative Knowledge (DL) r= 0.7358, GPA and Conditional Knowledge (CDL) with r= 0.6134, GPA and Planning (PL) with r = 0.7061, GPA and Information Management Strategies (IMS) with r = 0.6882, GPA and Comprehension Monitoring (CM) with r= 0.7025. A strong relationship also exists between GPA with Debugging (DBG) with r = 0.6023 and all are significant at the 1% level. This implies that each of the sub-component processes of the Knowledge about Cognition and Regulation of Cognition influence the academic achievement in the Introductory Computer Programming course at the University.

Table 4.5: Correlation Coefficient between GPA and other sub-components of MAI

Y	**X**	**r**	**r^2**	**t**	**Pr(>|t|)**
GPA	MAI	0.8226	0.6767	18.4136	0.0000
GPA	P	0.4387	0.1925	6.2140	0.0000
GPA	DL	0.7358	0.5413	13.8274	0.0000
GPA	CDL	0.6134	0.3762	9.8843	0.0000
GPA	PL	0.7061	0.4986	12.6917	0.0000
GPA	IMS	0.6882	0.4737	12.0747	0.0000
GPA	CM	0.7025	0.4935	12.5629	0.0000
GPA	DBG	0.6023	0.3627	9.6027	0.0000
GPA	EVL	0.5679	0.3225	8.7816	0.0000
GPA	KC	0.7483	0.5599	14.3562	0.0000
GPA	RC	0.8224	0.6763	18.3987	0.0000

4.2.4.1 Coefficient of Correlation between KC with sub-components of MAI

The results provided in Table 4.6 indicate that strong correlations exist between the Knowledge of Cognition (KC) with Procedural Knowledge (P) with r= 0.6698, Declarative Knowledge (DL) with r= 0.9071, Conditional Knowledge (CDL) with r= 0.8289, Planning (PL) with r= 0.7347, Comprehension Monitoring (CM) with r= 0.781. Moderate correlation existed between KC and Information Management Strategies (IMS) with r= 0.6548, Debugging (DBG) with r= 0.5446 and Evaluation with r= 0.6087. All are significant at the 1% level. From the results, it is clearly shown that the Declarative knowledge (DL) is highly correlated with Knowledge of Cognition (KC) with the coefficient of determination r^2 = 82% and the value of r= 0.9071 close to +1 indicates that if novice programmers possess higher knowledge of declarative, the Knowledge of Cognition can easily be acquired.

Table 4.6: Correlation Coefficient between Knowledge of Cognition (KC) and other sub-components of MAI

Y	X	r	r^2	t	Pr(>|t|)
KC	P	0.6698	0.4486	11.4797	0.0000
KC	DL	0.9071	0.8228	27.4249	0.0000
KC	CDL	0.8289	0.6870	18.8584	0.0000
KC	PL	0.7347	0.5398	13.7840	0.0000
KC	IMS	0.6548	0.4288	11.0268	0.0000
KC	CM	0.7810	0.6099	15.9145	0.0000
KC	DBG	0.5446	0.2966	8.2648	0.0000
KC	EVL	0.6087	0.3705	9.7643	0.0000

4.2.4.2 Coefficient Correlation between RC with sub-components of MAI

The test results presented in Table 4.7 show the coefficient correlation between the Regulation of Cognition (RC) with other sub-components of MAI. The Regulation of Cognition (RC) is strongly related to other sub-components of MAI as indicated in Table 4-6, which shows the coefficient correlation between Regulation of Cognition (RC) and Procedural Knowledge at r 0.55, Regulation of Cognition (RC) with Declarative Knowledge (DC) at r = 0.78, Regulation of Cognition (RC) with Conditional Knowledge (CDL) at r = 0.68, Regulation of Cognition (RC) with Planning (PL) at r = 0.86, Regulation of Cognition (RC) with Information Management Strategy (IMS) at r= 0.82, Regulation of Cognition (RC) with Comprehension Monitoring at r = 0.85, and, last, but not least, the Regulation of Cognition (RC) and Debugging Strategies (DBG) at r = 0.68.

Table 4.7: Correlation Coefficient between Regulation of Cognition (RC) and other sub-components of MAI

Y	X	r	r^2	t	Pr(>|t|)
RC	P	0.5523	0.3051	8.4328	0.0000
RC	DL	0.7878	0.6206	16.2789	0.0000
RC	CDL	0.6815	0.4644	11.8528	0.0000
RC	PL	0.8615	0.7423	21.6000	0.0000
RC	IMS	0.8199	0.6723	18.2308	0.0000
RC	CM	0.8521	0.7262	20.7263	0.0000
RC	DBG	0.6866	0.4714	12.0204	0.0000
RC	EVL	0.7451	0.5551	14.2175	0.0000

4.2.5 Descriptive Analysis

This section presents the descriptive analysis of the results obtained from the survey. This section is divided into eight themes/classifications of metacognitive skills, as defined by (Tobias & Everson, 1995). Respondents are grouped into four, with a different range of GPA scores for the Introductory Computer Programming subject (2.0 – 2.49, 2.5 – 2.99, 3.0 – 3.49 and 3.5 – 4.00). The reason behind this is to see the pattern (the GPA score with each metacognitive component) exhibited from the data collected.

4.2.5.1 Declarative knowledge (DL)

Declarative knowledge refers to the factual information and knowledge that a person possesses. It is one component that makes up the knowledge of Cognition within the MAI instrument. As described by (Tobias & Everson, 1995), there are six attributes or statements that made up the declarative knowledge component. The first attribute is about novices understanding their intellectual strengths and weaknesses. The second attribute is the awareness of knowing the kind of information that is most important to learn in respect of computer programming. Organizing information is the third attribute that makes up the declarative knowledge. The fourth attribute is the awareness of what the instructor expects novices to learn followed by the attribute concerning remembering information. The last attribute is the control over how well they learn the subject. Table 4.8 presents the comparison of declarative knowledge of the respondents based on their GPA in the computer programming subject.

The result shows that the higher degree of the declarative knowledge the better the student performance in the Computer Programming subject. Students with a higher degree of declarative knowledge know their weaknesses and strengths, know what to learn, are good at organizing information, good at remembering information and good at monitoring their knowledge. The ‘T’ symbol, as shown in Table 4.8, represents the ‘TRUE’ response whereas ‘F’ represents ‘FALSE’. The result obviously revealed that there is an increase in the response rate on ‘TRUE’ from the left to the right of the table, which gives an indication that the better students perform in the computer programming subject the better the declarative knowledge the student possesses.

From Table 4.8, it shows that 96% of the respondents with a GPA between 3.5 – 4.00 responded ‘TRUE’ to the attributes of that relates to their understanding of strength and weaknesses (first attributes), while the other 4% of the respondents from this group responded ‘FALSE’. For the group with a GPA of between 3.00 – 3.49, 97.5% responded ‘TRUE’ for the first attribute and 2.5% said ‘FALSE’. In addition, 94.63% of the group from GPA 2.5 – 2.99 responded ‘TRUE’ and 3.57% responded ‘FALSE’ for the first attribute.

The second attribute is about ‘Knowing what kind of information is most important to learn’. For the group with the GPA 3.5 – 4.00, 70% agreed and the other 30% disagreed with the second attribute. For the respondents with a GPA of between 3.0 and 3.49, 65% said ‘TRUE’ and 35% responded ‘FALSE’. While 73% of the respondents with a GPA between 2.5 and 2.99 agreed that they know the kind of information that is most important to learn, the other 27% did not agree. Most of the respondents (88%) with a GPA of 2.00 – 2.49 disagreed with the statement, only 12% from this group responded ‘TRUE’ for this statement.

The third attribute is about organizing information. From the results obtained in Table 4.8, it obviously revealed that almost all respondents (92%) with a good GPA (3.5 – 4.00) agreed that they are good at organizing information and only 8% did not agree; 70% said ‘TRUE’ and 30% said ‘FALSE’ from the group of respondents with a GPA between 3.0 – 3.49; and 36% responded ‘TRUE’ while 64% responded ‘FALSE’ from the group with a GPA between 2.5 and 2.99. For the group with a GPA of between 2.0 and 2.49, the majority (84%) said ‘FALSE’ while only 16% said ‘TRUE’.

The fourth attribute is about knowing what the instructor expects the student to learn. This clearly revealed that the better the student’s achievement in Introductory Computer Programming in terms of the GPA score, the better their skill of knowing what the instructor expects them to learn. A total of 79% of respondents with a GPA between 3.5 and 4.00 responded ‘TRUE’ for that statement while 21% responded ‘FALSE’; 82.5% said ‘TRUE’ while the other 17.5% said ‘FALSE’ for the group of respondents with a GPA between 3.0 and 3.49; 46% said ‘TRUE’ and 54% said ‘FALSE’ from the group of respondents with a GPA between 2.5 and 2.99, and for those respondents with a GPA between 2.0 and 2.49, 34% responded ‘TRUE’ and 66% responded ‘FALSE’.

The fifth attribute concerns remembering information, 71% of the respondents with a GPA between 3.5 and 4.00 responded ‘TRUE, while 29% responded ‘FALSE’; 70% of the respondents from the group with a GPA of 3.0 – 3.49 said ‘TRUE’, while 30% said

‘FALSE’. For the group of respondents with a GPA of 2.5 – 2.99, 50% said ‘TRUE’ and the other 50% said ‘FALSE’; 18% from the group with a GPA of 2.0-2.49 responded ‘TRUE’ and the majority with 82% said ‘FALSE’.

The last attribute pertains to the control of information. The collected data showed that students who performed well at the Introductory Computer Programming had a good control of information; 79% from the respondents group 3.5 – 4.00 responded ‘TRUE’ and 21% responded ‘FALSE’; 87.5% of respondents from the group with a GPA of between 3.0 and 3.49 responded ‘TRUE’ while 12.5% responded ‘FALSE’. For the group with a GPA of between 2.5 and 2.99, 43% responded ‘TRUE’ and 57% responded ‘FALSE’; 25% from the group of students with a GPA between 2.0 and 2.49 said ‘TRUE’ and 75% said ‘FALSE’. In this section, it can be concluded that the greater the declarative knowledge possessed by novices the better one’s performance in the computer programming subject.

Table 4.8: Comparison of ‘Declarative Knowledge’ by respondent’s GPA

Attr. No	***Declaration Knowledge Attribute***	**GPA**	**2.0 – 2.49**		**2.5 – 2.99**		**3.0 – 3.49**		**3.5 – 4.00**	
		True (T)/ False (F)	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**
1	I understand my intellectual strengths and weaknesses	**T**	33	75.00	54	96.43	39	97.50	23	95.83
		F	11	25.00	2	3.57	1	2.50	1	4.17
2	I know what kind of information is most important to learn	**T**	5	11.36	41	73.21	26	65.00	17	70.83
		F	39	88.64	15	26.79	14	35.00	7	29.17
3	I am good at organizing information.	**T**	7	15.91	20	35.71	28	70.00	22	91.67
		F	37	84.09	36	64.29	12	30.00	2	8.33
4	I know what the teacher expects to	**T**	15	34.09	26	46.43	33	82.50	19	79.17
		F	29	65.91	30	53.57	7	17.50	5	20.83

	learn.									
5	I am good at remembering information.	**T**	8	18.18	28	50.00	28	70.00	17	70.83
		F	36	81.82	28	50.00	12	30.00	7	29.17
6	I have control over how well I learn.	**T**	11	25.00	24	42.86	35	87.50	19	79.17
		F	33	75.00	32	57.14	5	12.50	5	20.83
7	I am a good judge of how well I understand something	**T**	5	11.36	31	55.36	31	77.50	21	87.50
		F	39	88.64	25	44.64	9	22.50	3	12.50
8	I learn more when I am interested in the topic.	**T**	33	75.00	53	94.64	36	90.00	22	91.67
		F	11	25.00	3	5.36	4	10.00	2	8.33

4.2.5.2 Procedural Knowledge (P)

This knowledge refers to how to perform a certain task, also known as imperative knowledge, which is another component of the knowledge of cognition within the MAI. Students that possess a higher degree of procedural knowledge know the appropriate strategy to use when studying. Not all respondents provide the information about their strategies in learning programming. However, some of the examples given by the respondents were as follows, ‘seek help of the Internet’, ‘trial and error method’, ‘seek the help of instructor’, and ‘study with others’. According to the Kolb' learning cycle (Harb, Durrant, & Terry, 1993), the most influential and effective idea of learning is the experiential, it is the learning by doing.

There are four attributes or statements that make up the procedural knowledge. The first attribute is about the past strategies used during learning computer programming, the second attribute is about student awareness of the specific strategies used for specific purpose, the third attribute is about the awareness of strategies used when studying and the last attribute is about the awareness of what helpful learning strategies to use when studying. Table 4.9 presents the comparison of the procedural knowledge group by CGPA. A total of 96% of the group of respondents with a GPA between 3.5 and 4.00 responded 'TRUE' for the first attribute, that they used strategies that have worked during learning, while only 4% responded 'FALSE'; 72.5% said 'TRUE' and 27.5% said 'FALSE' for the first attribute; this response rate comes from the group of respondents with a GPA between 3.0 and 3.49.

Table 4.9: Comparison of 'Procedural Knowledge' by respondent's GPA

Attr. No	***Procedural Knowledge Attribute***	**GPA**	**2.0 – 2.49**		**2.5 – 2.99**		**3.0 – 3.49**		**3.5 – 4.00**	
		True (T)/ False (F)	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**
1	I try to use strategies that have worked in the past	**T**	28	63.64	35	62.50	29	72.50	23	95.83
		F	16	36.36	21	37.50	11	27.50	1	4.17
2	I have a specific purpose for each strategy I use.	**T**	35	79.55	44	78.57	38	95.00	23	95.83
		F	9	20.45	12	21.43	2	5.00	1	4.17
3	I am aware of what strategies I used when I study.	**T**	30	68.18	40	71.43	38	95.00	23	95.83
		F	14	31.82	16	28.57	2	5.00	1	4.17
4	I find myself using helpful learning strategies automatically	**T**	24	54.55	36	64.29	29	72.50	21	87.50
		F	20	45.45	20	35.71	11	27.50	3	12.50

For respondents with a GPA between 2.5 and 2.99, 62.5% said 'TRUE' for this first attribute and 37.5% responded 'FALSE', and 63.6% of the respondents with a GPA between 2.0 and 2.49 responded 'TRUE' while the other 36.4% said 'FALSE'. For the second attribute (the awareness of a specific purpose for each strategy used), the majority of respondents (96%) from the group with a GPA between 3.5 and 4.00 responded 'TRUE' and 4% responded 'FALSE'. The same goes for the group with a

GPA of between 3.0 and 3.49, for which the majority of respondents (95%) responded ‘TRUE’, while the other 5% responded ‘FALSE’; 79% out of 56 respondents from the group with a GPA of between 2.5 and 2.99 said ‘TRUE’ and the other 21% said ‘FALSE’; 80% out of 44 respondents from the group with a GPA of 2.0 – 2.49 responded ‘TRUE’ and 20% responded ‘FALSE’. For the third attribute, which concerns the awareness of what strategies are used during studying, the majority of the respondents (96%) from the group with a GPA of between 3.5 and 4.00 responded ‘TRUE’ to the statement, while 4% did not agree with the statement and responded ‘FALSE’.

The same goes for the respondents with a GPA between 3.00 – 3.49, for which the majority (95%) responded ‘TRUE’ and 5% responded ‘FALSE’. While 71% out of 56 respondents from the group with a GPA of between 2.5 and 2.99 responded ‘TRUE’ and 29% responded ‘FALSE’. For the group of respondents with a GPA between 2.0 and 2.49, 54.6% responded ‘TRUE’, while 45.4% responded ‘FALSE’. The last attribute concerned the awareness of what helpful learning strategies are used when studying. The majority of the respondents (87.5%) with a GPA of between 3.5 and 4.00 agreed with the statement and responded ‘TRUE’, while the other 12.5% responded ‘FALSE’. For the respondents from the group with a GPA of between 3.00 and 3.49, the majority (72.5%) responded ‘TRUE’ and 27.5% responded ‘FALSE’. Out of 56 respondents from the group with a GPA of between 2.5 and 2.99, 64% responded ‘TRUE’ while the other 36% responded ‘FALSE’. It can be concluded from the results obtained, that the better the students performed in terms of the GPA obtained in Introductory Computer Programming the better the procedural knowledge they possessed.

4.2.5.3 Conditional Knowledge (CDL)

Conditional knowledge is a classification of the knowledge found in the long-term memory; it can be best described as knowing “when” and “why” to use declarative and procedural knowledge and ‘when’ not to (Schraw, 2006). In this study, there are five attributes or statements that make up the conditional knowledge. The first attribute is about a novice’s learning performance when they know something about the topic in computer programming. The second attribute is to answer the question: do novices apply different learning strategies for the different situations during learning computer programming? The third attribute is to know how far novices are self-motivated in learning programming. The fourth attribute is to know whether novices use their intellectual strengths to compensate for their weaknesses and the last attribute is to identify the novice’s learning effectiveness when applying each strategy during learning. Table 4.10 presents the comparison of conditional knowledge group by the respondent’s GPA in Introductory Computer Programming.

The majority of respondents from the group with a GPA between 3.5 and 4.00 agreed (by responding ‘TRUE’) with all five attributes, 87.5% for the first attribute, 100% for the second, third and fourth attribute and 87.5% for the last attribute; only 12.5% of the respondents from this group disagreed with the first and the last attribute. Similarly, for the respondents from the group with a GPA between 3.0 and 3.49, the majority responded ‘TRUE’ to all five attributes; 77.5% for the first attribute, 87.5% for the second attribute, 90% for the third attribute and 100% for the fourth attribute and 67.7% for the last attribute. Only 22.5% responded ‘FALSE’ for the first attribute, 12.5% for the second attribute, 10% for the third attribute and 32.5% for the last attribute. For respondents from the group with a GPA of between 2.5 – 2.99, 75% responded ‘TRUE’ for the first attribute, 87.5% responded ‘TRUE’ for the second and third attributes, 78.6% for the fourth attribute, and 77% for the last attribute. Conversely, 25% of the respondents from this group responded ‘FALSE’ for the first attribute, 12.5% responded ‘FALSE’ for the second attribute, 21.43% responded ‘FALSE’ for the third attribute and 23.21% responded ‘FALSE’ for the last attribute.

Out of 44 respondents from the group with a GPA of between 2.0 and 2.49, 34% responded ‘TRUE’ for the first attribute, 50% responded ‘TRUE’ for the second and fourth attribute, 68% responded ‘TRUE’ for the third attribute and 38.6% responded ‘TRUE’ for the last attribute. In total, 66% responded ‘FALSE’ for the first attribute, 50% responded ‘FALSE’ for the second and fourth attribute, and 61.6% responded ‘FALSE’ for the last attribute. It can be concluded in this section that the better that students performed in terms of the GPA obtained in the Introductory Computer Programming the better the conditional knowledge they possessed

Table 4.10: Comparison of ‘Conditional Knowledge’ by respondent’s GPA

Attr. No	***Conditional Knowledge Attribute***	**GPA**	**2.0 – 2.49**		**2.5 – 2.99**		**3.0 – 3.49**		**3.5 – 4.00**	
		True (T)/ False (F)	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**
1	I learn best when I know something about the topic.	**T**	15	34.09	42	75.00	31	77.50	21	87.50
		F	29	65.91	14	25.00	9	22.50	3	12.50
2	I use different learning strategies depending on the situation.	**T**	22	50.00	49	87.50	35	87.50	24	100.00
		F	22	50.00	7	12.50	5	12.50	0	0.00
3	I can motivate myself to learn when I need to.	**T**	30	68.18	49	87.50	36	90.00	24	100.00
		F	14	31.82	7	12.50	4	10.00	0	0.00
4	I use my intellectual strengths to compensate for my weaknesses.	**T**	22	50.00	44	78.57	40	100.00	24	100.00
		F	22	50.00	12	21.43	0	0.00	0	0.00

5	I know when each strategy I use will be most effective.	**T**	17	38.64	43	76.79	27	67.50	21	87.50
		F	27	61.36	13	23.21	13	32.50	3	12.50

4.2.5.4 Planning (PL)

Planning (also called forethought) is the process of thinking about and organizing the activities into a desired goal. Planning involves the creation and maintenance of a plan. As such, planning is a fundamental property of intelligent behavior. In this study, there are seven attributes that make up the planning knowledge. The first attribute concerns how the novice sets the pace while learning computer programming in order to have ample time. The second attribute is about students' realization of what they need to learn before beginning for any given task. The third attribute is about goal setting before the beginning any task. The fourth attribute concerns knowing how far the student prepares materials before learning, the fifth attribute relates to the way students solve problems in learning computer programming, The sixth attribute pertains to the things that novices must do right before the beginning of a task (read instruction) and the last attribute concerns how students manage their time in order to accomplish goals.

From Table 4.11, only 27% out of 44 respondents from the group with a GPA of between 2.0 – 2.49 responded 'TRUE' for the first attribute while the other 73% responded 'FALSE', for the second attribute, 35% responded 'TRUE', 65% responded 'FALSE'. Out of 44 respondents from this group 32% responded 'TRUE' for the third attribute and the other 68% responded FALSE, 41% responded 'TRUE' and 59% responded 'FALSE' for the fourth attribute. In addition, 47% responded 'TRUE' and 53 responded 'FALSE' for the fifth attribute; 34% out of 44 respondents from the group responded 'TRUE' for the sixth attribute and 64% responded 'FALSE'. While 20% from this group responded 'TRUE' for the last attribute and 80% responded 'FALSE';

64.3% out of 56 respondents from the group with GPA of between 2.5 and 2.99 responded ‘TRUE’ for the first attribute while the other 35.7% responded ‘FALSE’.

For the second attribute, 77% responded ‘TRUE’ and 23% responded ‘FALSE’. Out of 56 respondents from this group 71% responded ‘TRUE’ for the third attribute and the other 28% responded FALSE, 80% responded ‘TRUE’ and 20% responded ‘FALSE’ for the fourth attribute; 82% responded ‘TRUE’ and 18% responded ‘FALSE’ for the fifth attribute; 62.5% out of 56 respondents from the group responded ‘TRUE’ for the sixth attribute and 37.5% responded ‘FALSE’. In addition, 48% from this group responded ‘TRUE’ for the last attribute and 52% responded ‘FALSE’; 82.5% out of 40 respondents from the group with a GPA of between 3.0 and 3.49 responded ‘TRUE’ for the first attribute while the other 17.5% responded ‘FALSE’.

For the second attribute, 77.5% responded ‘TRUE’, 22.5% responded ‘FALSE’. Out of 40 respondents from this group the majority, 97.5%, responded ‘TRUE’ for the third attribute and only 2.5% responded FALSE, 85% responded ‘TRUE’ and 15% responded ‘FALSE’ for the fourth attribute. For the fifth attribute, 90% responded ‘TRUE’ and 10% responded ‘FALSE’; 95% out of 40 respondents from the group responded ‘TRUE’ for the sixth attribute and 5% responded ‘FALSE’ . While 87.5% from this group responded ‘TRUE’ for the last attribute and 12.5% responded ‘FALSE’; 79.2% out of 24 respondents from the group with a GPA of between 3.5 and 4.00 responded ‘TRUE’ for the first attribute while the other 20.8% responded ‘FALSE’.

For the second attribute, 91.7% responded ‘TRUE’, 8.3% responded ‘FALSE’. Out of 24 respondents from this group 95.8% responded ‘TRUE’ for the third attribute, and the other 4.2% responded FALSE, 91.7% responded ‘TRUE’ and 8.3% responded ‘FALSE’ for the fourth attribute; 95.8% responded ‘TRUE’ and 4.2% responded ‘FALSE’ for the fifth attribute; 87.5% out of 24 respondents from the group responded ‘TRUE’ for the sixth attribute and 12.5% responded ‘FALSE’. While 75% from this group responded ‘TRUE’ for the last attribute and 25% responded ‘FALSE’. It can be concluded that from the results obtained that the better the student performed in terms of the GPA obtained in Introductory Computer Programming the better the planning knowledge possessed by them

Table 4.11: Comparison of ‘Planning Knowledge’ group respondent’s GPA

Attr. No	*Planning Attribute*	**GPA**	**2.0 – 2.49**		**2.5 – 2.99**		**3.0 – 3.49**		**3.5 – 4.00**	
		True (T)/ False (F)	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**
1	I pace myself while learning in order to have enough time.	**T**	12	27.27	36	64.29	33	82.50	19	79.17
		F	32	72.73	20	35.71	7	17.50	5	20.83
2	I think about what I really need to learn before I begin a task.	**T**	15	34.09	43	76.79	31	77.50	22	91.67
		F	29	65.91	13	23.21	9	22.50	2	8.33
3	I set specific goals before I begin a task.	**T**	14	31.82	40	71.43	39	97.50	23	95.83
		F	30	68.18	16	28.57	1	2.50	1	4.17
4	I ask myself questions about the material before I begin.	**T**	18	40.91	45	80.36	34	85.00	22	91.67
		F	26	59.09	11	19.64	6	15.00	2	8.33
5	I think of several ways to	**T**	19	43.18	46	82.14	36	90.00	23	95.83
		F	25	56.82	10	17.86	4	10.00	1	4.17

	solve a problem and choose the best one.									
6	I read instructions carefully before I begin a task.	**T**	15	34.09	35	62.50	38	95.00	21	87.50
		F	29	65.91	21	37.50	2	5.00	3	12.50
7	I organize my time to best accomplish my goals.	**T**	9	20.45	27	48.21	35	87.50	18	75.00
		F	35	79.55	29	51.79	5	12.50	6	25.00

4.2.5.5 Information Management Strategy (IMS)

Table 4.12 depicts the distribution of respondents who responded concerning the attributes that relate to the information management strategies. The information management strategies skill refer to the skill of organizing information and the control over it, planning, structuring, elaborating and processing the information. It is clearly revealed that the respondents responded 'TRUE' for all the attributes increasing from left to the right of the table, which indicates that those who have a good achievement in computer programming (CGPA) have positive attributes towards the skill of information management strategies.

From Table 4.12, 61% out of 44 respondents from the group with a GPA of between 2.0 and 2.99 responded 'TRUE' for the first attribute while the other 17% responded 'FALSE', for the second attribute, 50% responded 'TRUE' and 'FALSE'. Out of 44 respondents from this group, 54% responded 'TRUE' for the third attribute and the other 45% responded FALSE, 52% responded 'TRUE' and 48% responded 'FALSE' for the fourth attribute. For the fifth attribute, 45% responded 'TRUE' and 55% responded 'FALSE', 71% out of 56 respondents from the group with a GPA of between

2.5 – 2.99 responded ‘TRUE’ for the first attribute while the other 29% responded ‘FALSE’. For the second attribute, 68% responded ‘TRUE’, 32% responded ‘FALSE’. Out of 56 respondents from this group 82% responded ‘TRUE’ for the third attribute and the other 18% responded FALSE, 64% responded ‘TRUE’ and 36% responded ‘FALSE’ for the fourth attribute.

For the fifth attribute, 75% responded ‘TRUE’ and 25% responded ‘FALSE’, 87.5% out of 40 respondents from the group with a GPA of between 3.0 and 3.49 responded ‘TRUE’ for the first attribute while the other 12.5% responded ‘FALSE’. For the second attribute, the majority of 90% responded ‘TRUE’ and only 10% responded ‘FALSE’. Out of 40 respondents from this group, the majority, with 82.5%, responded ‘TRUE’ for the third attribute and only 17.5% responded FALSE, 85% responded ‘TRUE’ and 15% responded ‘FALSE’ for the fourth and fifth attribute. In addition, 87.5% out of 24 respondents from the group with a GPA of between 3.5 and 4.00 responded ‘TRUE’ for the first attribute while the other 12.5% responded ‘FALSE’.

For the second attribute, 91.7% responded ‘TRUE’ and 8.3% responded ‘FALSE’. Out of 24 respondents from this group, 91.7% responded ‘TRUE’ for the third attribute and the other 8.3% responded FALSE, 83.3% responded ‘TRUE’ and 16.7% responded ‘FALSE’ for the fourth attribute. For the fifth attribute, 87.5% responded ‘TRUE’ and 12.5% responded ‘FALSE’. The comparison of the other attributes in IMS for the respondents is provided in Table 4.12.

Table 4.12: Comparison of 'Information Management Strategies' group by respondent's GPA

Attr. No	*Information Management Strategies Attribute*	GPA	2.0 – 2.49		2.5 – 2.99		3.0 – 3.49		3.5 – 4.00	
		True/ False	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**
1	I slow down when I encounter important information.	**T**	27	61.36	40	71.43	35	87.50	21	87.50
		F	17	38.64	16	28.57	5	12.50	3	12.50
2	I consciously focus my attention on important information.	**T**	22	50.00	38	67.86	36	90.00	22	91.67
		F	22	50.00	18	32.14	4	10.00	2	8.33
3	I focus on the meaning and significance of new information.	**T**	24	54.55	46	82.14	33	82.50	22	91.67
		F	20	45.45	10	17.86	7	17.50	2	8.33
4	I create my own examples to make information more meaningful.	**T**	23	52.27	36	64.29	34	85.00	20	83.33
		F	21	47.73	20	35.71	6	15.00	4	16.67
5	I draw pictures or diagrams to help me understand while learning.	**T**	20	45.45	42	75.00	34	85.00	21	87.50
		F	24	54.55	14	25.00	6	15.00	3	12.50
6	I try to translate new information into my own words.	**T**	23	52.27	45	80.36	37	92.50	22	91.67
		F	21	47.73	11	19.64	3	7.50	2	8.33
7	I use the organizational structure of the text to help me learn	**T**	22	50.00	40	71.43	34	85.00	23	95.83
		F	22	50.00	16	28.57	6	15.00	1	4.17
8	I ask myself if what I'm reading is related to what I already know	**T**	24	54.55	39	69.64	34	85.00	22	91.67
		F	20	45.45	17	30.36	6	15.00	2	8.33
9	I try to break studying down into smaller steps.	**T**	22	50.00	40	71.43	37	92.50	22	91.67
		F	22	50.00	16	28.57	3	7.50	2	8.33
10	I focus on	**T**	29	65.91	46	82.14	32	80.00	22	91.67

	overall meaning rather than specifics.	**F**	15	34.09	10	17.86	8	20.00	2	8.33

4.2.5.6 Comprehension Monitoring (CM)

Comprehension monitoring can be referred to as an assessment of the learning strategy one uses. There are seven sub-attributes-that made up the comprehension monitoring, as shown in Table 4.13. The first attribute is about the progress assessment monitoring of goal achievement, followed by attributes that relate to considering alternative solutions before attempting to solve the problem. The last three attributes are about analyzing the strategies used while studying and monitoring the knowledge obtained. From the left to the right view of Table 4.13, it can be seen that the higher the GPA of the respondents the better the comprehension monitoring skills possessed by them.

Only 27% out of 44 respondents from the group with a GPA between 2.0 and 2.49 responded 'TRUE' for the first attribute while the other 73% responded 'FALSE'; for the second attribute, 39% responded 'TRUE' and 61% responded 'FALSE'. Out of 44 respondents from this group 37% responded 'TRUE' for the third attribute and the other 63% responded FALSE; 41% responded 'TRUE' and 59% responded 'FALSE' for the fourth and fifth attribute; 32% of 44 respondents from the group responded 'TRUE' for the sixth attribute and 68% responded 'FALSE'; and 16% from this group responded 'TRUE' for the last attribute and 84% responded 'FALSE'. From the group with a GPA of between 2.5 and 2.99, 59% out of 56 respondents responded 'TRUE' for the first attribute while the other 41% responded 'FALSE'. For the second attribute, 68% responded 'TRUE', 32% responded 'FALSE'.

Out of 56 respondents from this group, 80% responded ‘TRUE’ for the third attribute and the other 20% responded FALSE; 82.14% responded ‘TRUE’ and 18% responded ‘FALSE’ for the fourth attribute; 66% responded ‘TRUE’ and 34% responded ‘FALSE’ for the fifth attribute; 66% out of 56 respondents from the group responded ‘TRUE’ for the sixth attribute and 34% responded ‘FALSE’; while 59% from this group responded ‘TRUE’ for the last attribute and 41% responded ‘FALSE’. From the group with a GPA of between 3.0 and 3.4987.5%, 40 respondents responded ‘TRUE’ for the first attribute while the other 12.5% responded ‘FALSE’. For the second attribute, 92.5% responded ‘TRUE’ and 7.5% responded ‘FALSE’. Out of 40 respondents from this group 100% responded ‘TRUE’. In addition, 95% responded ‘TRUE’ and 5% responded ‘FALSE’ for both the fourth and fifth attribute; 90% out of 40 respondents from the group responded ‘TRUE’ for the sixth attribute and 10% responded ‘FALSE’; 85% from this group responded ‘TRUE’ for the last attribute and 15% responded ‘FALSE’; 83% out of 24 respondents from the group with a GPA of between 3.5 and 4.00 responded ‘TRUE’ for the first attribute while the other 17% responded ‘FALSE’. For the second attribute, 91.7% responded ‘TRUE’, 8.3% responded ‘FALSE’.

Out of 24 respondents from this group 87.5% responded ‘TRUE’ for the third attribute and the other 12.5% responded FALSE, 83% responded ‘TRUE’ and 17% responded ‘FALSE’ for the fourth and fifth attribute; 92% responded ‘TRUE’ and 8% responded ‘FALSE’ for the sixth attribute and 12.5% responded ‘FALSE’; while 75% from this group responded ‘TRUE’ for the last attribute and 25% responded ‘FALSE’. It can be concluded that from the results obtained that the better students performed in terms of the GPA obtained in Introductory Computer Programming the better comprehension monitoring skill possessed by them.

Table 4.13: Comparison of 'Comprehension Monitoring' by respondent's GPA

Attr. No	*Comprehension Monitoring Attribute*	**GPA**	**2.0 – 2.49**		**2.5 – 2.99**		**3.0 – 3.49**		**3.5 – 4.00**	
		True (T)/ False (F)	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**
1	I ask myself periodically if I am meeting my goals.	**T**	12	27.27	33	58.93	35	87.50	20	83.33
		F	32	72.73	23	41.07	5	12.50	4	16.67
2	I consider several alternatives to a problem before I answer.	**T**	17	38.64	38	67.86	37	92.50	22	91.67
		F	27	61.36	18	32.14	3	7.50	2	8.33
3	I ask myself if I have considered all options when solving a problem.	**T**	16	36.36	45	80.36	40	100.00	21	87.50
		F	28	63.64	11	19.64	0	0.00	3	12.50
4	I periodically review to help me understand important relationships.	**T**	18	40.91	46	82.14	38	95.00	20	83.33
		F	26	59.09	10	17.86	2	5.00	4	16.67
5	I find myself analyzing the usefulness of strategies while I study.	**T**	18	40.91	45	80.36	38	95.00	20	83.33
		F	26	59.09	11	19.64	2	5.00	4	16.67
6	I find myself pausing regularly to check my comprehension.	**T**	14	31.82	37	66.07	36	90.00	22	91.67
		F	30	68.18	19	33.93	4	10.00	2	8.33
7	I ask myself questions about how well I am doing while learning something new.	**T**	7	15.91	33	58.93	34	85.00	19	79.17
		F	37	84.09	23	41.07	6	15.00	5	20.83

4.2.5.7 Debugging Strategies (DBG)

Debugging is one of the important skills that need to be developed by novice programmers. This knowledge refers to the ability to identify errors and correct them. In this study, there are five attribute/statements that relate to the debugging strategies attributes. The first attribute is about novices' agreement or disagreement that they seek help from others when they do not understand something. The second attribute is about students' agreement or disagreement that they change their strategies when they fail to understand something during learning. The third attribute is about re-evaluation of strategies and assumption when getting confused about something during learning programming. The fourth and fifth attribute relate to the repetitive process of something that is unclear and confused (e.g. re-read).

Table 4.14: Comparison of 'Debugging Strategies' by respondent's GPA

Attr. No	***Debugging Strategies Attributes***	**GPA**	**2.0 – 2.49**		**2.5 – 2.99**		**3.0 – 3.49**		**3.5 – 4.00**	
		True/ False	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**
1	I ask others for help when I don't understand something	**T**	13	29.55	45	80.36	28	70.00	24	100.00
		F	31	70.45	11	19.64	12	30.00	0	0.00
2	I change strategies when I fail to understand.	**T**	17	38.64	48	85.71	34	85.00	21	87.50
		F	27	61.36	8	14.29	6	15.00	3	12.50
3	I re-evaluate my assumptions when I get confused.	**T**	23	52.27	42	75.00	35	87.50	23	95.83
		F	21	47.73	14	25.00	5	12.50	1	4.17
4	I stop and go back over new information that is not clear.	**T**	20	45.45	49	87.50	32	80.00	20	83.33
		F	24	54.55	7	12.50	8	20.00	4	16.67
5	I stop and reread when I get confused.	**T**	12	27.27	44	78.57	31	77.50	21	87.50
		F	32	72.73	12	21.43	9	22.50	3	12.50

Table 4.14 presents a comparison of the debugging strategies group by GPA. All 100% of respondents from the group with GPA between 3.5 and 4, say ‘TRUE’ that they seek help from others when they do not understand about something during learning, 87.5% agree on the second attributes by responding ‘YES’ that they will change strategies when they fail to understand something during learning while the other 12.5 responded ‘FALSE’, 96% responded true for the third attribute and only 4% responded ‘FALSE’, 83% responded ‘TRUE’ and 17% responded ‘FALSE’ for the fourth attribute, and 87.5% responded ‘TRUE’ for the last attribute and 12.5% responded ‘FALSE’.

For the group of respondents with a GPA between 3.00 and 3.49, 70% out of 40 respondents responded ‘TRUE’ and 30% responded ‘FALSE’ for the first attribute, 85% responded ‘TRUE’ for the second attribute and 15% responded ‘FALSE’; 87.5% responded ‘TRUE’ that they re-evaluate the assumption when get confused while 12.5% responded ‘FALSE’, while 80% from this group of respondents responded ‘TRUE’ and 20% disagreed by responding ‘FALSE’. Out of 56 respondents from the group with a GPA of between 2.5 and 2.99, 80% responded ‘TRUE’ and 20% responded ‘FALSE’ to the first attribute, 86% responded ‘TRUE’ to the second attribute and 14% responded ‘FALSE’, 75% responded ‘TRUE’ and 25% responded ‘FALSE’ to the third attribute, 87.5% responded ‘TRUE’ and 12.5% responded ‘FALSE’ to the fourth attribute and 79% responded ‘TRUE’ to the last attribute and 21% responded ‘FALSE’. For the group of respondents with a GPA of 2.0 – 2.49, 30% responded ‘TRUE’ that they seek help from others when they don’t understand something while 70% responded ‘FALSE’, 86% agreed with the second attribute by responding ‘TRUE’. From analysis of the results obtained, it can be concluded that the better students performed in

Introductory Computer Programming the better the procedural knowledge possessed by them.

4.2.5.8 Evaluation (EVL)

Evaluation refers to appraising the final task and the efficiency in which the task was performed. This can include re-evaluating strategies that were used after the learning episode. In this study, there are six attributes that make up the evaluation attribute. The first attribute concerns the self-evaluation after the test. The second attribute is about assessment of the easiest way to execute things after finishing a task. The third attribute is about a student's ability to see what they have learned in a conceptual view. The fourth attribute concerns the student's ability to assess how well the goal is accomplished. The fifth attribute is about the student's ability to assess all the available options after solving the problem. The last attribute is about self-evaluation concerning how much one has learned once they have completed a given task. Table 4.15 presents the comparison of evaluation knowledge group by GPA. Out of 44 respondents from the group with a GPA of between 2.0 and 2.49, 34% responded 'TRUE' and 66% responded 'FALSE' for the first attribute, 50% responded 'TRUE' and 50% responded 'FALSE' for the second attribute, 45% responded 'TRUE' and 55% responded 'FALSE' for the third attribute, 57% responded 'TRUE' and 43% responded 'FALSE' for the fourth attribute, 43% responded 'TRUE' and 57% responded 'FALSE' for the fifth attribute, and, for the last attribute, 45% of the respondents from this group responded 'TRUE' and 56% responded 'FALSE'.

From the total of 56 respondents from the group with a GPA of between 2.5 and 2.99, 34% responded 'TRUE' and 66% responded 'FALSE' for the first attribute and second attribute, 79% responded 'TRUE' and 21% responded 'FALSE' for the third attribute, 70% responded 'TRUE' and 30% responded 'FALSE' for the fourth attribute, 62.5% responded 'TRUE' and 37.5% responded 'FALSE' for the fifth attribute, 43% responded 'TRUE', and, for the last attribute, 59% of the respondents from this group responded 'TRUE' and 41% responded 'FALSE'. From the total of 40 respondents from the group with a GPA between 3.0 and 3.49, 65% responded 'TRUE' and 35% responded 'FALSE' for the first attribute, 82.5% responded 'TRUE' and 17.5% responded 'FALSE' for the second attribute, 80% responded 'TRUE' and 20% responded 'FALSE' for the third attribute, the majority with 90% responded 'TRUE' and only 10% responded 'FALSE' for the fourth attribute, 80% responded 'TRUE' and 20% responded 'FALSE' for the fifth attribute, and, for the last attribute, 55% of the respondents from this group responded 'TRUE' and 45% responded 'FALSE'. Out of 24 respondents from the group with a GPA of between 3.0 – 3.49. The majority of the respondents with 92% responded 'TRUE' and only 8% responded 'FALSE' for the first four attributes, 87.5% responded 'TRUE' and 12.5% responded 'FALSE' for the fifth and the last attribute.

Table 4.15: Comparison of 'Evaluation' by respondent's GPA

Attr. No	***Evaluation Attributes***	**GPA**	**2.0 – 2.49**		**2.5 – 2.99**		**3.0 – 3.49**		**3.5 – 4.00**	
		True (T)/ False (F)	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**	**Freq**	**Perct (%)**
1	I know how well I did once I finish a test.	**T**	15	34.09	37	66.07	26	65.00	22	91.67
		F	29	65.91	19	33.93	14	35.00	2	8.33
2	I ask myself if there was an easier way to do things after I finish a task.	**T**	22	50.00	37	66.07	33	82.50	22	91.67
		F	22	50.00	19	33.93	7	17.50	2	8.33
3	I summarize what I've learned after finishing	**T**	20	45.45	44	78.57	32	80.00	22	91.67
		F	24	54.55	12	21.43	8	20.00	2	8.33
4	I ask myself	**T**	25	56.82	39	69.64	36	90.00	22	91.67

	how well I accomplished my goals once I'm finished	**F**	19	43.18	17	30.36	4	10.00	2	8.33
5	I ask myself if I have considered all options after I solve a problem	**T**	19	43.18	35	62.50	32	80.00	21	87.50
		F	25	56.82	21	37.50	8	20.00	3	12.50
6	I ask myself if I learned as much as I could have once I finished a task.	**T**	20	45.45	33	58.93	22	55.00	21	87.50
		F	24	54.55	23	41.07	18	45.00	3	12.50

4.2.6 Types of Strategy in Learning Programming

Learning strategies refer to any behavior or thought that facilitates encoding in such a way that knowledge integration and retrieval are enhanced (Weinstein, 1988). These thoughts and behaviors constitute organized plans of action designed to achieve a certain goal (Weinstein & Mayer, 1983). Table 4.16 presents the results of the type of strategies used by novices to deal with the perceived difficulties in learning computer programming. As shown in table 4.20, 'Help-seeking' and 'Critical-Thinking' are the two predominant strategies chosen by respondents with 80% of the responded rate followed by 'peer-learning' with a frequency of 123 out of 164 respondents and a response percentage of 75%. In 'Help-seeking' type of strategy, students read a book to learn more about a topic but if they are confused about the topic they will re-read and ask help from the instructor or someone expert in that area. Whereas in 'Critical Thinking' strategy, students try to solve the problem and distinguish the differences about a topic to learn. In this study, 'Peer learning' and 'Help-seeking' are categorized as help-seeking approaches, as described by (Newman, 2002). Apart from that, 60% of respondents said that 'Metacognition Regulation' is a strategy used to deal with the difficulties in learning computer programming.

Table 4.16: Types of strategy in learning programming

Strategy types	**Sub Strategy**	**Example**	**Frequency**	**Percentage (%)**
Rehearsal	Rehearsal	"Read the text again and go over notes." "Print some slides I have not already printed – do vocabulary words and definitions."	33	21
Meaningful	Elaboration	"Study over them and make easy to remember examples." "Read more carefully in the textbook and relate material to my own experience/knowledge."	38	23
	Organization	"read chapter, review notes, make a chart of information" "Read over notes and draw some diagrams. Use some memory tricks and apply to my background knowledge."	74	45
	Critical Thinking	"read over them more to distinguish the differences among them." "work problem-solving" Study the schedules of reinforcement."	131	80
Self-regulatory	Metacognition Regulation	"I definitely need to learn my stages of thinking and reasoning. I need to define Piaget and Vygotsky/ likes vs. differences." "Read it more thoroughly and try to apply it to different situations (quiz myself)"	98	60
	Time and Study Environment	"Try to go over more concepts more in my own time." "Preview before class"	74	45
	Effort regulation	"I will go over at home and study correlations and try to	74	45

		figure out problems and focus on group activity." "I will read the chapters involved with the lecture and make note cards covering the main topics to organize the info."		
Help Seeking	Peer Learning	"Read the book and discuss with peers." "I will go home and read over the material more and if I need more help I will either ask another student or the professor."	123	75
	Help Seeking	"I will read more about it in the book and if I am still confused I will ask for help." "Reread the chapter, look at the specifics, ask instructor if any questions after re-reading."	131	80

In this type of strategy, students learn by applying topic learning to the different situations. In addition, 40% of respondents used a 'Time and Study Environment'. Students who select this strategy usually use more time to learn more concepts in the topic to learn and have a preview before class. These two strategies, 'Metacognition Regulation' and 'Time and study environment' fall under Self-regulatory strategy. In contrast, 45% of respondents responded to the 'Organization' and 'Effort Regulation' strategy. In the 'Organization' type of strategy, students usually translate the information on the chapter or topic learned with the use of a chart or diagram to reflect their understanding and apply to their background knowledge. Furthermore, 23% of the respondents responded to 'Elaboration' as a learning strategy in which the student usually studies the topic to learn and makes it easy to remember examples (e.g. short notes) and tries to relate it to their experience or knowledge. Only 21% of the respondents responded to 'Rehearsal'. In this type of strategy, the student makes notes and does vocabulary on words and definitions

4.2.7 Motivational Factors

Table 4.17 shows the results of the motivating factors that will encourage students in utilizing support system in learning computer programming. The explanations of the results are given in the next paragraph.

Table 4.17: Motivating Factors of using support environment

Attributes	**Definition**	**Frequency**	**(%)**
Efficiency	The degree to which users feel that the software assists them in learning (Kirakowski & Corbett, 1988) (Kirakowski & Corbett, 1988)	98	60
Affect	The degree to which users feel that the software assists them in learning (Kirakowski & Corbett, 1988) ((Kirakowski & Corbett, 1988)	115	70
Effort	The degree to which the software is self-explanatory, as well as more specific things like the adequacy of help facilities and documentation (Kirakowski, 1998)	98	60
Goal Orientation	Individuals primarily strive to enhance their knowledge, skills, and competence, referred to as a learning orientation, or generally attempt to demonstrate their abilities and expertise, referred to as a performance orientation	98	60
Helpfullness	The degree to which the software is self-explanatory, as well as more specific things like the adequacy of help facilities and documentation (Kirakowski, 1998)	131	80
Learnability	Measures the speed and facility with which the user feels that they have been able to master the system, or to learn how to use new features when necessary (Kirakowski, 1998)	115	70
Control	Measures the extent to which the user feels in control of the software, as opposed to being controlled by the software, when carrying out the task (Kirakowski, 1998)	98	60
Self-regulation	Integrated learning process, consisting of the development of a set of constructive behaviors that affect one's learning.	115	70
User-friendly	Refers to anything that makes it easier for novices to use a computer. Menu-driven programs, for example, are considered more user-friendly than command-driven systems. Graphical user interfaces (GUIs) are also considered user-friendly. Online help systems are another feature of user-friendly programs.	164	100

From the results obtained, all the respondents (100%) are motivated to use a support system if it provided user-friendly features that made it easy to use the system. On the other hand, 80% preferred to have a 'Helpfulness' feature to motivate them to use the system. This 'Helpfulness' attribute refers to the degree to which the software is self-explanatory, as well as more specific things like the adequacy of help facilities and documentation. Approximately, 70% of the respondents are motivated if the system is able to provide the 'Affect', 'Learnability' and 'Self-Regulation' kind of features. 'Affect' refers to the degree to which users feel that the software assists them in learning. 'Learnability' can be defined as the speed and facility with which the user feels that they have been able to master the system, or to learn how to use new features when necessary, whereas 'Self-Regulation' refers to the Integrated learning process, consisting of the development of a set of constructive behaviors that affect one's learning. While 60% of the respondents are motivated to use the system that provides the attributes 'Efficiency', 'Effort', 'Goal Orientation' and 'Control'.

4.3 Further Analysis

Table 4.18 presents the multiple linear regression results. The results revealed that Declarative Knowledge (DL) and Information Management Strategy (IMS) are the two strong subcomponents of MAI that contribute significantly to the effect of a student's learning success in an introductory computer programming course at the University at the 1% level. The Conditional Knowledge is another subcomponent of MAI that makes a significant contribution towards the learning success at the 5% level.

Table 4.18: Multiple linear regression result

Attribute	Coef.	std	t(155)	p-value
Intercept	1.435311	0.111444	12.879183	0.000000
P	0.018408	0.030892	0.595889	0.552118
DL	0.062143	0.021997	2.825044	0.005350
CDL	0.053789	0.028191	1.907979	0.058243
PL	0.019846	0.025259	0.785687	0.433250
IMS	0.064113	0.017725	3.617059	0.000403
CM	0.030684	0.025590	1.199069	0.232331
DBG	0.025224	0.028729	0.877980	0.381313
EVL	0.017236	0.022784	0.756505	0.450495

4.4 Reliability Test of Survey Result

The alpha coefficient for the eight items of metacognition knowledge (Planning, Declarative knowledge, Conditional knowledge, Information Management Strategies, Comprehension Monitoring, Debugging and Evaluation) (see Table 4.19) is .912 (see Table 4.20), suggesting that the items have a relatively high internal consistency. (Note that a reliability coefficient of .70 or higher is considered "acceptable" in most social science research situations (Santos, 1999)).

Table 4.19: Item-Total Statistics

Attr.	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Squared Multiple Correlation	Cronbach's Alpha if Item Deleted
P	33.43	102.824	.494	.266	.918
DL	31.52	79.540	.804	.708	.893
CDL	32.70	95.107	.679	.510	.905
PL	31.79	78.083	.856	.777	.887
IMS	29.07	83.872	.693	.523	.904
CM	31.71	78.390	.834	.787	.890
DBG	32.52	90.889	.755	.633	.899
EVL	32.45	90.973	.691	.492	.903

Table 4.20: Reliability Statistics

Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
.912	.915	8

The column named ‘Cronbach's Alpha if Item Deleted’ presents the value that Cronbach's alpha would be if that particular item was deleted from the scale. We can see that removal of any question, except question P (Planning), would result in a lower Cronbach's alpha. Therefore, we would not want to remove these questions. Removal of question P (Planning) would lead to a small improvement in Cronbach's alpha, and we can also see that the "Corrected Item-Total Correlation" value was low (0.494) for this item. This might lead us to consider whether we should remove this item.

4.5 Discussion of the Survey

The empirical research work examined the relationship between metacognition and student achievement in the Introductory Computer Programming course. It was realized that metacognitive awareness has a significant effect on the successful learning of introductory computer programming at universities. The conducted experiment answered the question, ‘Do metacognitive skills have an effect on students’ learning success in introductory computer programming at university?’ From this research work, the findings confirmed that metacognition has a positive effect on students’ introductory programming learning success at university and the results indicate that most of MAI subcomponents make a positive contribution towards the learning success. The test also shows that Knowledge of Cognition (KC) and Regulation of Cognition (RC) have a strong positive correlation, given the conclusion that the way students plan, strategic learning, monitor learning, correct comprehension errors and evaluate their learning have an impact on what learners know about their conceptual knowledge and vice versa. This study proved that the higher the degree of metacognitive awareness possessed by novice programmers the greater the learning success in Introductory Computer Programming at university. From the study, it was also found that that two subscales or subcomponents of MAI that have been identified as strong contributors to learning success are Declarative Knowledge (DL) and Information Management Strategy (IMS). As discussed in the literature, declarative knowledge is the factual knowledge the learner needs before being able to process or use critical thinking related to the topic.

The finding of this empirical study is similar to the results of other studies (Camahalan, 2006; Donker, De Boer, Kostons, van Ewijk, & Van der Werf, 2014; Young & Fry, 2012; Zimmerman & Kitsantas, 2014). In his research, (Camahalan, 2006) states that "students with given of appropriate or favorable time or occasion to think metacognitively and clearly expressed their thought of metacognitive strategies are more likely to be affected positively in their academic achievement." With the given positive relationship result between MAI score and GPA, MAI can be a tool used by lecturers to screen students that need assistance to improve their metacognitive skills in learning computer programming. This may become useful when lecturers have little opportunity to know their students individually, especially in large classes as well as online classes. The summary of the survey can be drawn as follows:

- The analysis of the result shows that students that performed well in Introductory Computer Programming (Indicated by the GPA) possess good declarative knowledge and vice versa. Students that are lack declarative knowledge have a problem in recognizing their intellectual strength and weaknesses, are unable to digest information that is important when studying, lack organizing information, and are weak at remembering information and knowledge understanding judgement.
    - *Solution*: Provide a system that can trigger reflection in novices in order to make them realize the benefits of general strategies and available resources, as well as the degree of attention that is necessary to succeed in the problem-solving process and the activity

- The analysis of the results show that students with good procedural knowledge show good achievement in Introductory Computer Programming as indicated by the GPA obtained. Students that lack procedural knowledge, conditional knowledge and information management strategies are not aware of what strategies can be used and under what circumstances specific processes or skills should transfer and are unable to process information efficiently.
    - *Solution*: Provide a system that can make them reflect on available strategies in solving problems. This activity would trigger and help them to think of the strategies that are relevant to the problem and apply them appropriately
- Students that have the ability in comprehension monitoring and debugging strategies are performed well in Introductory Computer Programming compared to those who lack these two skills. Students that lack comprehension monitoring and debugging strategies are unable to evaluate their understanding as they read and correct the problem and they are also unable to use appropriate strategies to correct comprehension and performance errors.
    - *Solution*: Provide a system that can reflect their understanding on the concept of the given problem as well as their confidence to solve the problem correctly by solving the given problem and present the answer.
- In this study, the findings show that students that have the ability to evaluate their knowledge performed well in Introductory Computer Programming, as indicated by the GPA obtained. Students that lack evaluation knowledge are unable to analyze their own performance and strategy effectiveness after a learning episode

    - *Solution*: Provide a system that includes the activity of checking answers and provides a library of solutions that are similar to the problem given for students to refer to, and evaluate their understanding.

- Overall, it can be concluded that those who possess all eight knowledge skills – declarative knowledge, procedural knowledge, conditional knowledge, planning, debugging strategies, information management strategies, evaluation, comprehension monitoring – that make up the metacognitive awareness, as described by (Tobias & Everson, 1995), have the potential to succeed in computer programming as proven in this study. Thus, providing a system to stimulate this knowledge is considered to be a valuable effort in computer science education.
- 'Help-seeking' and 'Critical-Thinking' are the two predominant types of strategy used in dealing with the difficulty in learning programming. Emphasis will be given to the features to support these two strategies during the design and implementation of the proposed system.
- The results show that the majority of the students are motivated to use the support system if it provides criteria like 'user friendly', 'helpfulness', 'affect', 'learnability' and 'self-regulation'
    - Solution: Provide a system that offers the required features for novices in learning computer programming metacognitively.

4.6 Interview Findings

This section is divided into two sections. The first section presents the findings of interviews with expert lecturers with the main objective being to investigate metacognitive implementation and metacognition awareness in teaching the computer programming course at universities. The second section discusses the findings of the interviews with the novice programmers with the objective being to gain further understanding concerning the learning behavior of the novice programmer in learning computer programming, which would help in refining the support features of the proposed system.

4.6.1 Interview Findings with Expert Lecturers

Data from the interview sessions were transcribed verbatim and analyzed. Each participant conducted a transcript review before it was analyzed. The results of the interviewing sessions are presented in Appendix E.

4.6.2 Demographic Study

Five expert lecturers from local universities participated in this study (3 males and 2 females). All the selected respondents are aged between 35 and 55 years. Three of them are PhD holders and the others are master degree holders. The participation was voluntary in nature and each interview session lasted around 45 minutes. “Expert lecturers” in this study refers to someone who has more than 5 years’ experience in teaching various types of programming language and paradigms, such as JAVA and structured programming with C++ or C, and who are actively involved in research projects and consultation regarding artificial intelligence, software engineering or parallel processing.

4.6.3 Expert Lecturer Definition of Metacognition

In the beginning of the interviewing session, all the expert lecturers were asked whether they had heard the term ‘metacognition’. All the expert lecturers’ responded that they had never heard about ‘metacognition’. A piece of paper presenting a definition of ‘metacognition’ was provided to each of them. The definition of ‘Metacognition’ is presented as follows:

“Metacognition is a form of critical thinking that enables understanding, analysis and control of one’s cognition, especially when engaged in teaching and learning. It can take many forms that include knowledge about when and how to use particular strategies for learning or for problem-solving. It is about ‘cognition about cognition’ or ‘knowing about knowing”(Metcalfe & Shimamura, 1994; Schraw, 1998).

After reading the definition of metacognition, they were asked again what they understand about the terms. One of them responded to “applying learning theory” while others answered that metacognition is “the way learners learn and develop knowledge”, “Awareness of Cognition” with only one respondent giving an almost complete answer, that is, “Awareness of Cognition and Monitoring of Cognition”. The results indicate that most of the participants responded to “Awareness of cognition” and “Monitoring Cognition”. Hence, it is feasible to discuss the appropriate definition of metacognition with expert lecturers

4.6.4 The role of Metacognition in teaching Computer Programming

All participants agreed that metacognition is one important factor in teaching success. Respondent 1 who had almost 15 years teaching C++ programming said that “Lecturers must think hard, as it is important for effective learning”. While other responses were as follows, “It is important and lecturers must think about their own thinking”, “Lecturers must think hard, it is important for effective learning”, “Educators must understand their own teaching processes”, “Lecturers must be aware of their own thoughts, emotions, and behaviors”

4.6.5 Metacognition application before teaching Computer Programming

All participants responded to “Planning”. The answers given by the respondents to the question are “Design and plan lessons”, “lesson-plans”, “Prepare for class the day before and arrive in the classroom a few minutes early”, “Analyze the teaching material and confirm the content of the lesson”, “write outline for discussion in classroom” and “make notes and review activities”. All the activities stated above are the activities that take place

4.6.6 Metacognition application during teaching Computer Programming

All participants responded that the ‘student’ is the important variable in teaching. The example of respondents were as follows, “prepared well before teaching, monitor myself during teaching to ensure message can be conveyed effectively to the student”, “listening to student and trying to answer any questions in simplest way that can be easily understood”, “interaction with student” “writing,” “discussing,”, “questioning and thinking with students”, “problem-solving”, “book and journal reading”, “spotting obstacles”, “mapping concept”, ”modeling”, “direct instruction”, “presenting”, “thinking aloud”, “reciprocal teaching”.

4.6.7 Metacognition application after teaching Computer Programming

The highest response from all participants was “Evaluating” or “Assessing”. The examples from the respondents were as follows “Prepares for the following session and tries to add further information on content that was not sufficiently clarified in the previous session”, “Evaluating the strength and weakness of one’s own teaching”, “Assessing/evaluating student learning by giving short quizzes”, “assessing teaching and planning for next round teaching” and “remains in the classroom for 15 or 30 minutes after class to answer questions from students". One of the respondents indicated that he reflects by asking himself "How do I feel about my teaching? Am I satisfied with my teaching? Where do I feel satisfied or unsatisfied? Why do I have this feeling? Where and how do I need to revise my lesson plan for the next session?" Another respondent said that he views teaching in three stages, before entering the classroom the instructor must plan the lesson, during teaching the instructor must act and think, and, after teaching, the instructor must reflect.

4.6.8 Summary of the Interview with Expert Lecturer

All the respondents were aware of metacognition. The study found that expert lecturers acknowledge and recognize the role of metacognition in teaching and learning in computer programming. Fourteen metacognition strategies employed during teaching were identified. These include coaching, questioning, mapping concept, problem-solving, reciprocal teaching, thinking aloud, reading books, modeling, journal, presenting, direct instruction, writing, asking to think and discussing. An efficient lecturer must understand himself, must be aware of each student and know the content of the knowledge as well as the pedagogical approach applied during teaching; this is to ensure that the knowledge can be conveyed effectively.

4.6.9 Interview Findings with Novice Programmers

As mentioned in Chapter three, the main objective of this study is to get further understanding on novices learning behavior in computer programming learning in order to gain a better insight of the support features for the suggested system. The recorded results were transcribed verbatim and analyzed. The respondents who participated in the survey were chosen from the undergraduate students that took the programming subject.

Out of ten invitations by email, only five responded and agreed to participate in this study. All of the respondents share the same background in that they have no experience in programming. The interviewing session took place at FSKM, Universiti Teknologi MARA (UiTM) Shah Alam. Appendix F provides the responses given by the respondents. The questions that were asked during the session are as follows:

1. Do you know your own strengths and weakness in learning programming?
2. How do you motivate yourself to learn programming?
3. What are the types of resource that you usually refer to?
4. What are the techniques used to process information in learning programming?
5. What kind of help seeking method do you use to help you understand while learning programming?
6. Do you track the amount of time taken for studying programming individually? How long will you spend for the revision?
7. How do you evaluate your own knowledge performance and understanding?

For the first question, most of the respondents did not recognize their own strengths and weaknesses in learning programming. Only respondent R4 was aware of his strengths and weaknesses. As explained in the literature study, what the learner knows and does not know is related to the declarative knowledge as well as the 'Person Variables'. Falvell (1979) indicates that it is important for one to be able to process or use his/her critical thinking that is related to the learning topic.

Question 2 is about how they are motivated to learn programming. R1, R2 and R5 all responded to 'cooperative learning', such as knowledge sharing with peers to make them motivated to learn programming. R3 claimed that programming is a difficult subject and he is not motivated to learn programming at all. In contrast, R4 is self-motivated and enjoys learning programming. Question 3 is about the main resources of reference for learning programming. The Internet is the predominant resource of reference for all respondents to learn programming. Apart from the Internet, books and materials given by the instructor are also the resources used by novices as references.

Question 4 relates to the information management strategies to process information. It also relates to the ‘Strategies Variable’, as described by Flavell (1979), the strategies applied to understanding the knowledge. Respondent R1 translates the information obtained into the words that can be understood by him, whereas R3 uses the notes summarization technique to process information. Respondents R2 and R4 use the picture and diagram to represent the information or knowledge obtained whereas respondent R5 uses organizational structure to represent the information.

Question 5 also relates to the Strategies Variable; R1 uses the social media as the medium to seek help while learning; to seek help from the social media like forums and Facebook; to seek the help of the instructor; seek the help of the instructor or someone that has a good knowledge of programming; study with friends and comparing answers from the quizzes, tests and assignments given by lecturer; find similar problem solutions from the Internet and by trial and error.

Question 6 pertains to the time consumed for learning computer programming individually. Most of the respondents did not record the time taken to learning programming individually. Only R1 responded that he spent about 2- 4 hours for learning programming individually.

Question 7 is about how students evaluate their own knowledge performance and understanding. R1 responded that he often communicates with the lecturer in order to know his strengths and weaknesses. R2 familiarizes himself with exercises and seeking help from the instructor when confused. R3 and R4 have almost similar answers – check quizzes, tutorials and test papers and comparing with instructor or others. R5 uses the Internet to check the answers of tests, exam papers or quizzes for similar problems.

4.6.10 Summary of Interviews with Novices

Based on the above study, several conclusions can be drawn as follows:

- Most of the respondents were motivated to learn programming if there is 'Cooperative learning' such as peer-learning and discussion
    - Proposed feature: Provide a forum where the student can share their knowledge and thoughts.
- There is no proper medium for students to make reference in learning programming and the Internet is the main resource for the novice programmer to make reference:
    - Proposed a support system for them to learn programming.
- The various skills and strategies of process information show that respondents are aware of their metacognitive skills:
    - Proposed a support system that can explicitly stimulate their metacognitive thinking in learning programming.

4.7 Summary

Consistent with other researchers (Donker et al., 2014; Scheier, Carver, Clark, & Fiske, 2014; Schraw & Dennison, 1994; Young & Fry, 2012), there was a significant correlation between regulation of the cognition factor and the knowledge of the cognition factor. Correlations were also found between the MAI and academic achievement. The regulation of the cognition factor of the MAI correlated with the GPA and the same is true for the knowledge of the cognition factor. The results of the survey helps researchers to identify the relevant activities to be embedded in the proposed support system. The summary of the survey section is stated as follows:

- Students that lack declarative knowledge have a problem in recognizing their intellectual strength and weaknesses, unable to digest information that is important when studying, lack of organizing information, weak at remembering information and knowledge understanding judgement:
    - *Solution*: Provide a system that can trigger the reflection of novices in order to make them realize the benefits of general strategies, available resources as well as the degree of attention that is necessary to succeed in the problem-solving process and the activity
- Students that lack procedural knowledge, conditional knowledge and information management strategies are not aware of what strategies should be used and under what circumstances specific processes or skills should be transferred and are unable to process information efficiently:
    - *Solution*: Provide a system that can make them reflect on the available strategies for solving problems. This activity would trigger and help them to think of the strategies that are relevant to the problem and apply them appropriately

- Students that lack comprehension monitoring and debugging strategies are unable to evaluate their understanding as they read and correct the problem, and they are also unable to use appropriate strategies to correct comprehension and performance errors:
    - *Solution*: Provide a system that can reflect their understanding on the concept of the given problem as well as their confidence to solve the problem correctly by solving the given problem and presenting the answer.
- Students that lack evaluation knowledge are unable to analyze their own performance and strategy effectiveness after a learning episode:
    - *Solution*: Provide a system that includes the activity of checking answers and providing a library of solutions that are similar to the problem given for students to refer to, and evaluate their understanding.
- Overall, it can be concluded that those who possess all eight knowledge skills – the declarative knowledge, procedural knowledge, conditional knowledge, planning, debugging strategies, information management strategies, evaluation, and comprehension monitoring – that make up the metacognitive awareness, as described by (Tobias & Everson, 1995), have the potential to succeed in computer programming, as proven in this study. Thus, providing a system to stimulate such knowledge is considered as a valuable effort in computer science education
- Expert lecturers apply metacognition in teaching, as well as teaching metacognition skills to the students to improve their learning in computer programming. Expert lecturers recognize the importance of metacognition in their teaching and computer science education, Metacognition is a learning

success factor of computer programming that must be done before, during and after instruction:

- Solution: Provide a system that includes metacognitive activities that are covered before, during and after the learning process

Based on the findings from the interviews, three stages have been identified to present the metacognitive activities within the MSSNP, before, after and during the teaching and learning. As suggested by (Thiede, Griffin, Wiley, & Redford, 2009; Zimmerman, 1990), the moment in time where the system provides metacognitive instruction is an important consideration. The findings of the interviews with expert lecturer helps researchers to design the activities to support the development of metacognition skill that takes place before, during and after teaching and learning, as shown in Table 4.21.

Table 4.21: Metacognitive stages and activities

Stages	**Description**	**Activity Example**
Before teaching and learning activity	Self-reflection that takes place before the learning process leads to the potential to put the learner in the correct condition or frame of mind to perform the task	- Planning - Goal setting - Selecting strategies - Identify potential obstacles
During Teaching and learning	Self-reflection occurs during the learning process and can help the student in the self-monitoring process.	- Actual cognitive activities - Spotting obstacles - Spotting problems in solving cognitive problem
After Teaching and learning	Self-reflection that happens after the learning process is a natural time for the student to reflect on their learning process and performance.	- Evaluation - Judging efficiency of the plan and the execution of a plan (Assessing the teaching effectiveness)

Based on the conducted survey and interviews, we propose five stages of the MSSNP, namely, Pre-task, Familiarization, Production, Evaluation and Post-Task.

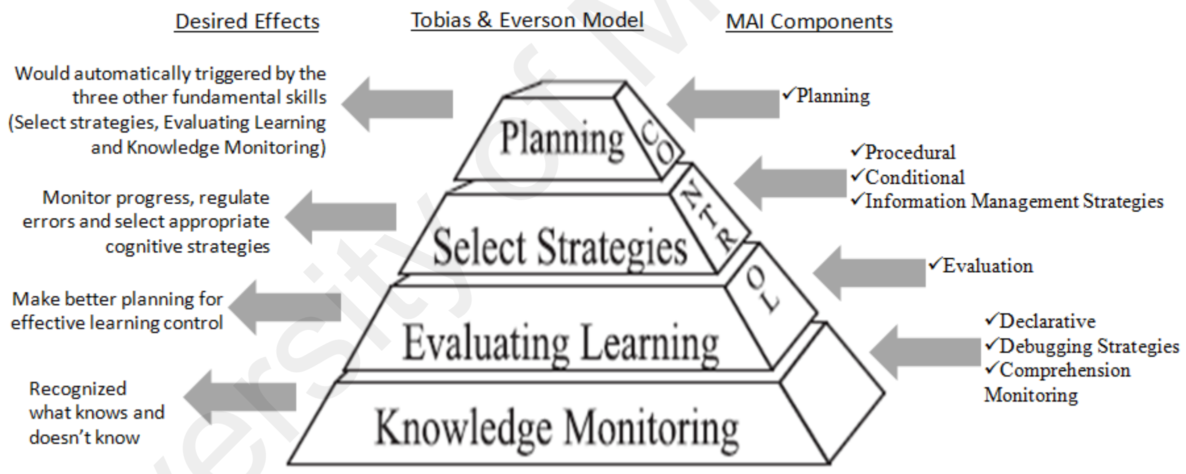
These five stages of the MSSNP are derived from the identified elements in supporting learning metacognitively, such as (should take place before, during and after learning process), the two components of knowledge that made up the metacognitive skills (Knowledge of Cognition and Regulation of Cognition) and the eight components that made up these two types of knowledge (Knowledge of Cognition and Regulation of Cognition) named as procedural knowledge, declarative knowledge, conditional knowledge, planning, information management strategies, comprehension monitoring, debugging strategies and evaluation. The mapping summarization of all these three elements in producing the MSSNP stages is presented in Table 4.22 and the desired effects for the MSSNP are shown in Figure 4.5.

Table 4.22: The mapping of time, components of metacognition, metacognition sub-components with the proposed stages in the MSSNP

Time	Components of Metacognition	Sub-components of metacognition	Proposed stages in the MSSNP
Pre- learning Activity **(**Self-reflection that takes place before the learning process provides the potential to put the learner in the correct condition or frame of mind to perform the task)	**Knowledge of Cognition** (The reflective aspect of learning that corresponds to what novices know about themselves, strategies, and conditions under which strategies are most useful. Declarative, procedural, and conditional knowledge can be thought of as the building blocks of conceptual knowledge)	**Procedural Knowledge** (How to perform task)	**Pre-Task** The objective of this stage is to trigger reflection on the student knowledge monitoring progress. It focuses on the past problem and performance (low, average or high) of the student and comparing their estimation and judgment to their actual knowledge understanding. Students will be provided with suitable conditions for making them realize the benefits of the general strategies, available resources as well as the degree of attention that is necessary to succeed in the problem-solving process and the activity • Knowledge

			monitoring state (KMA applied (From the Concept of Tobias and Everson)
		Declarative knowledge (Information that a person knows)	**Familiarization** The objective of this phase is to make novice programmers reflect on strategies as this activity helps them identify the strategies that are relevant and apply them appropriately. This phase focuses on metacognitive strategies related to the process during problem-solving
Actual learning Activity (Cognitive Strategy Instruction Approach (CSI)) **(**Self-reflection occurs during the learning process and can help the student in the self-monitoring process.)		**Conditional Knowledge** (Knowledge found in long-term memory)	
			Production The objective of this stage is to reflect the student's understanding concerning the concept as well as their confidence in solving the problem correctly.
	Regulation of Cognition **(**The term 'Regulation of Cognition' refers to the Control aspect of learning that corresponds to knowledge about the way students plan, implement strategize, monitor, correct comprehension errors, and evaluate their learning)	**Planning** (goal setting, allocating resources)	
		Information Management Strategies (This is related to the skills and strategy sequences used to process information more efficiently (e.g. organizing, elaborating, summarizing, selective focusing))	
		Comprehension Monitoring **(**Assessment one's learning or strategy used)	**Evaluation** This stage only involves the activity of checking the solution provided by the lecturer, which is used as a comparison in studying the student's solution
Post-Learning Activity **(**Self-reflection happens after the learning process is a natural time for the student to		**Debugging Strategies** **(**Strategies used to correct comprehension and performance errors)	
		Evaluation	**Post-Task**

reflect on their learning process and performance.)		(Analysis of performance and strategy effectiveness after a learning episode)	The activities designed in this stage provide the opportunity to the student to review their most recent experience, and explore what happened during the problem-solving activity. The aim is to assist the student in identifying the 'cause of the mistake' that relates to the problem, the resources used and the issues relating to time management.



Desired Effects
Tobias & Everson Model
MAI Components
Would automatically triggered by the three other fundamental skills (Select strategies, Evaluating Learning and Knowledge Monitoring)
Monitor progress, regulate errors and select appropriate cognitive strategies
Make better planning for effective learning control
Recognized what knows and doesn't know
Planning
Select Strategies
Evaluating Learning
Knowledge Monitoring
CONTROL
✓Planning
✓Procedural
✓Conditional
✓Information Management Strategies
✓Evaluation
✓Declarative
✓Debugging Strategies
✓Comprehension Monitoring

Figure 4.6: Theoretical model and the proposed desired effects for the MSSNP

CHAPTER 5: DESIGN AND IMPLEMENTATION OF METACOGNITIVE SUPPORT SYSTEM FOR NOVICE PROGRAMMERS (MSSNP)

The analysis of the survey and interviews that were conducted resulted in the actual requirements of a metacognitive learning environment for novice programmers in learning computer programming. The requirements were then translated into the architectural design of a Metacognitive Support System for Novice Programmers (MSSNP). Figure 5.1 illustrates the proposed architectural design of a metacognitive support system for learning computer programming activities. Semantic Web is the underlying technology for supporting all the designed activities in the prototype system.

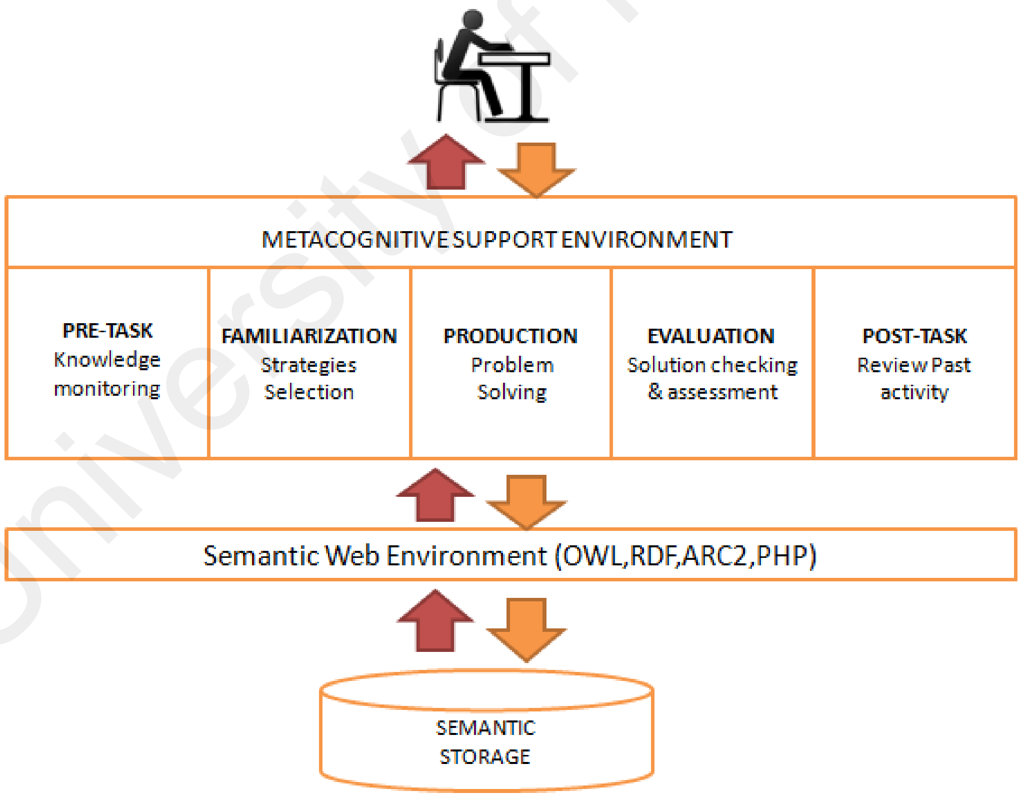

METACOGNITIVE SUPPORT ENVIRONMENT
PRE-TASK Knowledge monitoring
FAMILIARIZATION Strategies Selection
PRODUCTION Problem Solving
EVALUATION Solution checking & assessment
POST-TASK Review Past activity
Semantic Web Environment (OWL,RDF,ARC2,PHP)
SEMANTIC STORAGE

Figure 5.1: Proposed architectural design of the MSSNP

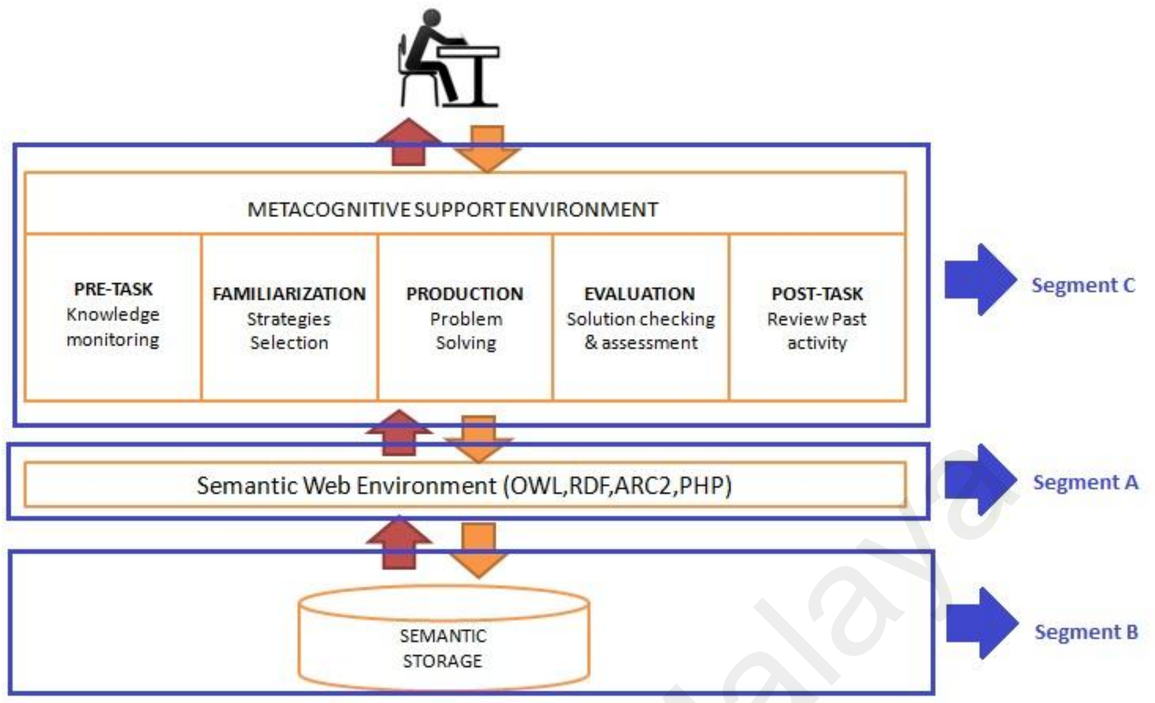

METACOGNITIVE SUPPORT ENVIRONMENT
PRE-TASK Knowledge monitoring
FAMILIARIZATION Strategies Selection
PRODUCTION Problem Solving
EVALUATION Solution checking & assessment
POST-TASK Review Past activity
Segment C
Semantic Web Environment (OWL,RDF,ARC2,PHP)
Segment A
SEMANTIC STORAGE
Segment B

Figure 5.2: The decomposition of process by segments in developing the MSSNP

We have decomposed the architectural design of the MSSNP into three segments, named as Segment A, Segment B and Segment C (as illustrated in Figure 5.2) in order to accomplish a clear understanding of the process involved in designing and constructing the MSSNP. The process of realizing the system starts with segment A where the process involves the MSSNP Ontology being designed and used as the backbone of the MSSNP system in order to process the content related to the system. The MSSNP Ontology is used to facilitate machine interpretability of content that is supported by XML, RDF and RDF D by providing additional vocabulary along with a formal semantic. The process involved in segment A is discussed further in section 5.2. Segment B involves the process of developing the triplestore data storage, also known as Semantic storage. The process involved in segment B is discussed in section 5.3. The last segment in this research work is Segment C. In this stage, the model, system framework, flow process, and software architecture design take place. The discussion of the process involved in this segment is provided in section 5.4.

5.2 Segment A- Ontology Construction

Ontological analysis clarifies the structure of knowledge. Given a domain, its ontology forms the heart of any system of knowledge representation for that domain. As discussed in Chapter 2, the Semantic Web technology is employed for the following reasons:

- Shared understanding of metacognitive skills and facilitate their development
- Organize activities in cognitive skill and metacognitive skill
- Promises a powerful approach to satisfy the E-learning requirement (e.g. material semantically annotated according to user preference)

Without ontologies, or the conceptualizations that underlie knowledge, there cannot be a vocabulary for representing knowledge. Thus, the first step in devising an effective knowledge representation system and vocabulary, is to perform an effective ontological analysis of the field, or domain. Weak analyses lead to incoherent knowledge bases. Ontologies are used in an increasing range of applications, notably the Semantic Web, and essentially have become the preferred modeling tool. However, the design and maintenance of ontologies is a formidable process. In this study, Protégé is the software tool used for authoring the MSSNP. This software is supported by a strong community of academic, corporate and government users to build a knowledge-based solution. Protégé fully supports the latest OWL 2 Web Ontology Language and RDF. In this study, we define seven steps to develop the MSSNP Ontology as follows:

Step 1. Define Scope and boundary (Discussed in Section 5.2.1)

Step 2. Reuse consideration (Discussed in Section 5.2.2)

Step 3. Class design/enumerate terms (Discussed in Section 5.2.3)

Step 4. Taxonomic identification (Discussed in Section 5.2.4)

Step 5. Property identification (Discussed in Section 5.2.5)

Step 6. Data property identification (Discussed in Section 5.2.6)

Step 7. Anomaly validity check (Discussed in Section 5.2.7)

5.2.1 Define Scope and Boundary

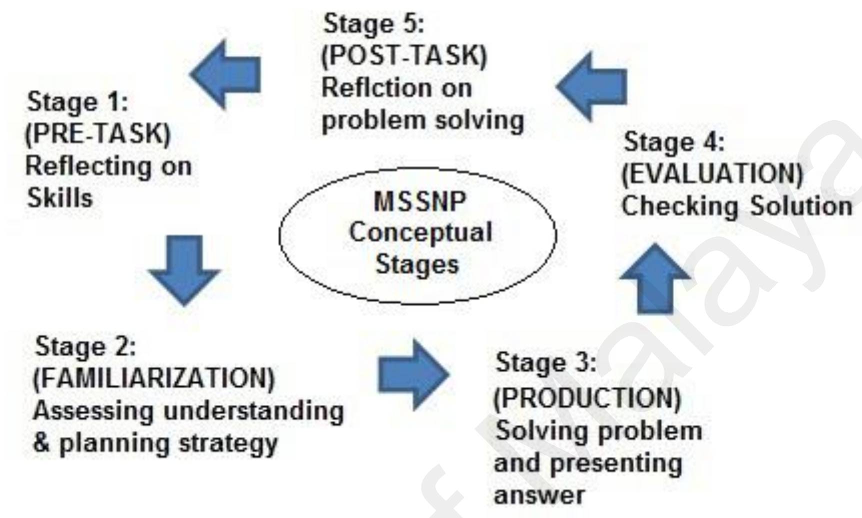


Figure 5.3: The MSSNP Conceptual Stages

Defining the scope of the ontology gives clear boundaries in respect of what should and should not be included in the Ontology. The scope will also determine the granularity of the ontology. One of the ways of determining the scope of the ontology is by sketching a list of questions, competency questions (Grüninger & Fox, 1995) concerning how to achieve its objectives. Later, the litmus test of using these questions will be used to determine whether the information depicted in the ontology is sufficient to correct the responses to these questions. Is a very specific representation of an answer needed. These competency questions do not need to be comprehensive as they are just a sketch. Following the MSSNP framework (Figure 5.3), the definition of each state and its objective with the possible competency questions at each stage is provided as follows:

- **Pre-Task** Stage, also called the Reflective Stage, caters for the metacognitive aspect that is necessary in solving a new problem. To make students reflect on what they know and do not know, this activity takes place before the process of learning happens and pertains to self-reflection. The appropriate conditions will be presented to the student to make them realize the importance of using suitable and possible strategies, resources that are available for them to refer to as well as the degree of focus that is necessary to succeed in solving a problem. The main objective of this stage is for students to be triggered by their own reflection in monitoring their knowledge. It focuses on the past experience in solving problems as well as the performance that is indicated by low, average or high. Students will be able to compare their estimation and judgment of their knowledge with the actual understanding through exposure to the following activities
    - Knowledge monitoring and performance comparison
    - Analysis of knowledge monitoring

  The competency questions that could possibly be derived from this stage are:
    - What is the student's performance in learning?
    - What strategies should be used?
    - What are the appropriate resources to be used?
    - How should the student judge their own knowledge?
    - How difficult are the questions?
- **Familiarization** stage is where students assess their understanding and strategy planning for problem-solving. During this stage, students are presented with a list of possible strategies for them to select and apply the most appropriate for solving the problem, alternatively they can compose the new strategies themselves. The primary objective of this stage is to make them reflect on the

strategies that may help them solve the problem. This activity emphasizes the metacognitive strategies that relate to the process of solving the problem. Students will be exposed to the activities that allow them to compare their own knowledge judgment with their actual knowledge and helps them to select the strategies that are provided to them to solve the problem. The possible competency questions that could be derived from this stage are:

  - What is the most suitable strategy to apply in solving the problem?
  - How can knowledge monitoring be measured?
  - How much time is needed for solving the problem?

- **Production** stage is the stage for students to do self-assessment and problem comprehension. In this stage, students are required to solve the given problem by presenting the answer. The main objective of this stage is to reflect the students' understanding concerning the concept as well as their confidence in solving the problem correctly. This activity relates to the actual assessment of student performance. The activities that take place during this stage are concentrate on translating the problem given into the pseudocode and monitoring the application of their planned strategies. The activities involved in this stage comprises of problem-solving, checking answer and quiz. The possible competency questions for this stage are as follows:
  - What is the correct answer for a particular problem?
  - How do you solve a problem?
  - How difficult is the problem?
- **Evaluation** stage is the stage where students can evaluate their experience of past problem-solving. This stage only involves the activity of checking the solution provided by the lecturer, which is used as a comparison in studying the student's solution.

- **Post-Task stage** takes place after the evaluation stage the main objective of which is for the student to reflect on the problem solved. The opportunity will be given to the students to review their most recent experiences as well as explore things that happened during the activity of solving the problem. Students will be able to identify the ‘cause of the mistakes’ that occur during the problem-solving activity, the time spent and the resources used. The self-questioning that will help students during this stage for triggering their learning experience are as follows:
    - Did I use the available resources in solving the problem?
    - How did I spend the time in solving the problem?
    - What is the possible plan of action to be used?

5.2.2 Reuse consideration

Ontology re-use is an agreed upon goal in ontology engineering. It reduces the cost of creating ontologies, improves the quality of the resulting ontologies, and eases later interaction between systems. The re-use of ontologies and of knowledge collected in the context of ontology creation comes in many forms Ontologies may be referenced, imported, taken as a starting point for extensions and revisions, or taken as a templates for the development of similar ontologies in other domains or for other purposes. However In this research work, the ontology is created from scratch since there is a lack of ontology in E-learning supporting the metacognitive learning environment. Considering what someone else has contributed can be worthwhile, in that we can extend the existing sources of particular tasks and domains by refining them to suit a particular scenario. Reusing existing sources may be a requirement for a system that needs to communicate with other applications that have already committed to a specific vocabulary that is controlled by the organization, field or domain.

5.2.3 Class design/enumerate terms

Class is one of the types of construct in OWL that allow us to represent knowledge about the domain. As discussed in section 5.2.1, the competency question derived from the scope and the boundary helped us in enumerating terms that could be used for building the classes for the MSSNP ontology. It is beneficial to list down all the terms that we intend to use to explain things to a user. Taking into consideration the properties of the terms; for example, in this study, important metacognitive learning related terms will include the student, syllabus, performance, knowledge monitoring, strategy and so on. First and foremost, it is important to comprehensively list down all the related terms, concepts or classes. The next step is to develop the association, class hierarchy between classes and defined properties. Based on the conceptual stages of the MSSNP, the identification of possible terms for each stage is presented in Table 5.1.

Table 5.1: Part of Classes used in the MSSNP Ontology

MSSNP Stages	**Class Name**	**Class Description**
PRODUCTION	AnswerExplanation	Class to describe the answer
PRE-TASK FAMILIARIZATION PRODUCTION EVALUATION POST-TASK	Lecturer	Class to represent a lecturer
FAMILIARIZATION PRODUCTION EVALUATION POST-TASK	LibraryProblem	Class that annotates the library of problems
PRE-TASK FAMILIARIZATION PRODUCTION EVALUATION POST-TASK	MetacognitiveStrategy	Class of Metacognitive Strategies
PRODUCTION EVALUATION POST-TASK	Performance	Class of range of performance and indicator
PRODUCTION EVALUATION	Problem	Class of problem and question
PRODUCTION EVALUATION	ProblemMultipleAnswer	Class of multiple answers for a problem

PRODUCTION EVALUATION	ProblemSolution	Class of problem solutions
PRE-TASK FAMILIARIZATION PRODUCTION EVALUATION POST-TASK	ProgrammingStrategy	Class to represent the strategy in learning programming
PRE-TASK FAMILIARIZATION PRODUCTION EVALUATION POST-TASK	Student	Class for Student
PRODUCTION EVALUATION POST-TASK	StudentPerformance	Class of student performance in the form of KMA
PRODUCTION EVALUATION POST-TASK	TimeInterval	Class that is used to capture the Time used by student to complete an answer

AnswerExplanation is a class that describes the answer for a particular problem. The Class occurs during the PRODUCTION stage and is located in the library of solutions used by the student for reference. StudentPerformance is the class that is used to capture student performance in the form of KMA Value. This is used during the PRODUCTION, EVALUATION and POST-TASK stages. The MetacognitiveStrategy class represents the metacognitive strategies used by the student to solve a given problem. This class occurs in all the MSSNP stages, that is, during the PRE-TASK, FAMILIARIZATION, PRODUCTION, EVALUATION and POST-TASK. This is similar for other classes such as LibraryProblem, Student and ProgrammingStrategy. LibraryProblem is a class that is used to annotate the library of problems. Whereas ProgrammingStrategy is a class that is used to represent the strategy in learning programming. The sub-concept of Programming strategies is presented in Figure 5.4 and Figure 5.5.

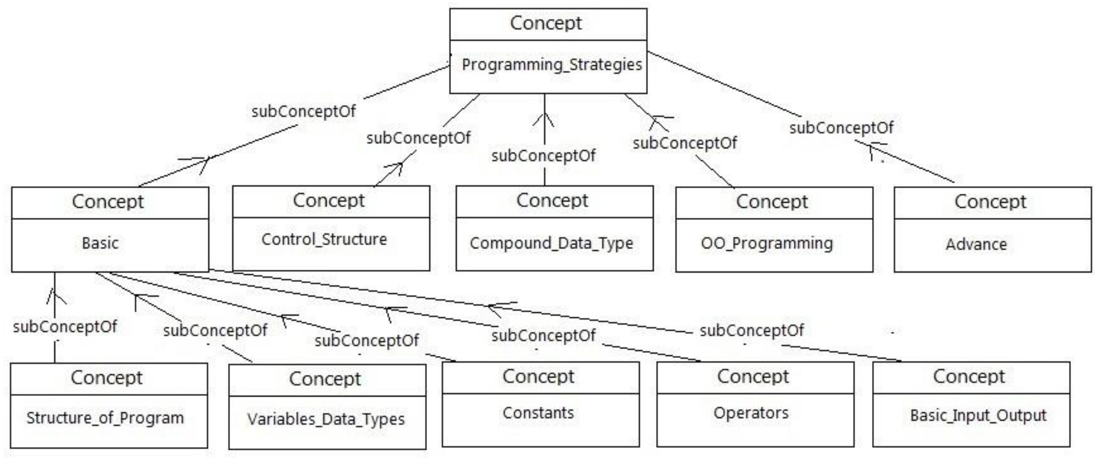


Figure 5.4: Basic Programming Strategy

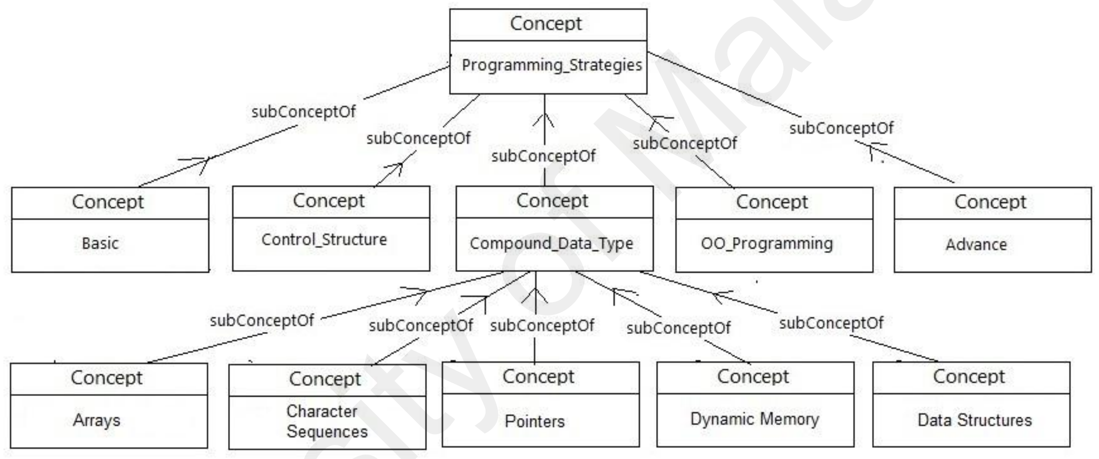


Figure 5.5: Compound Data Type programming strategy

The diagrams presented in Figure 5.4 and Figure 5.5 are translated into the MSSNP Ontology and written in Web Ontology language (OWL). The value of this built-in OWL property is a list of individuals that are the instances of the class. This enables a class to be described by exhaustively enumerating its instances. A class description of the "enumeration" kind is defined with the owl:oneOf property. An example of part of the enumeration applied in the MSSNP Ontology is as follows:

```
<owl:Class>
  <owl:oneOf rdf:parseType="operators">
    <owl:Thing rdf:about="#Assignment"/>
    <owl:Thing rdf:about="#Arithmethic"/>
    <owl:Thing rdf:about="#Increment"/>
    <owl:Thing rdf:about="#Decrement"/>
    <owl:Thing rdf:about="#Relational"/>
    <owl:Thing rdf:about="#Comparison"/>
    <owl:Thing rdf:about="#Comma"/>
    <owl:Thing rdf:about="#Bitwise"/>
    <owl:Thing rdf:about="#ExplicitTypeCasting"/>
    <owl:Thing rdf:about="#sizeof"/>
    <owl:Thing rdf:about="#precedence"/>
  </owl:oneOf>
</owl:Class>
```

5.2.4. Taxonomic identification

Ontology implies a broader scope of information. People often use taxonomies as a tree. An ontology might encompass a number of taxonomies with each taxonomy organizing a subject in a particular way. In this study, we have employed the combination of top-down and bottom-up. Bottom-up and top-down are two opposite approaches to developing a hierarchical structure. In the top-down approach, the broadest terms are identified first and then narrower terms are selected to reach the desired level of specificity. While the bottom-up approach occurs when a list of terms have been derived from a corpus of content. In this study, we generalized and specialized them appropriately after specifying the key concepts. We started with the top-level approach, such as programming strategies, before connecting them to a middle-level concept, such as object oriented programming and drill down to polymorphism as the specialization chapters. According to (Noy & McGuinness, 2001), none of these approaches (A top-down, bottom-up, combination) is superior to any of the others. The technique to choose strongly depends on the personal view of a particular domain (Noy & McGuinness, 2001).

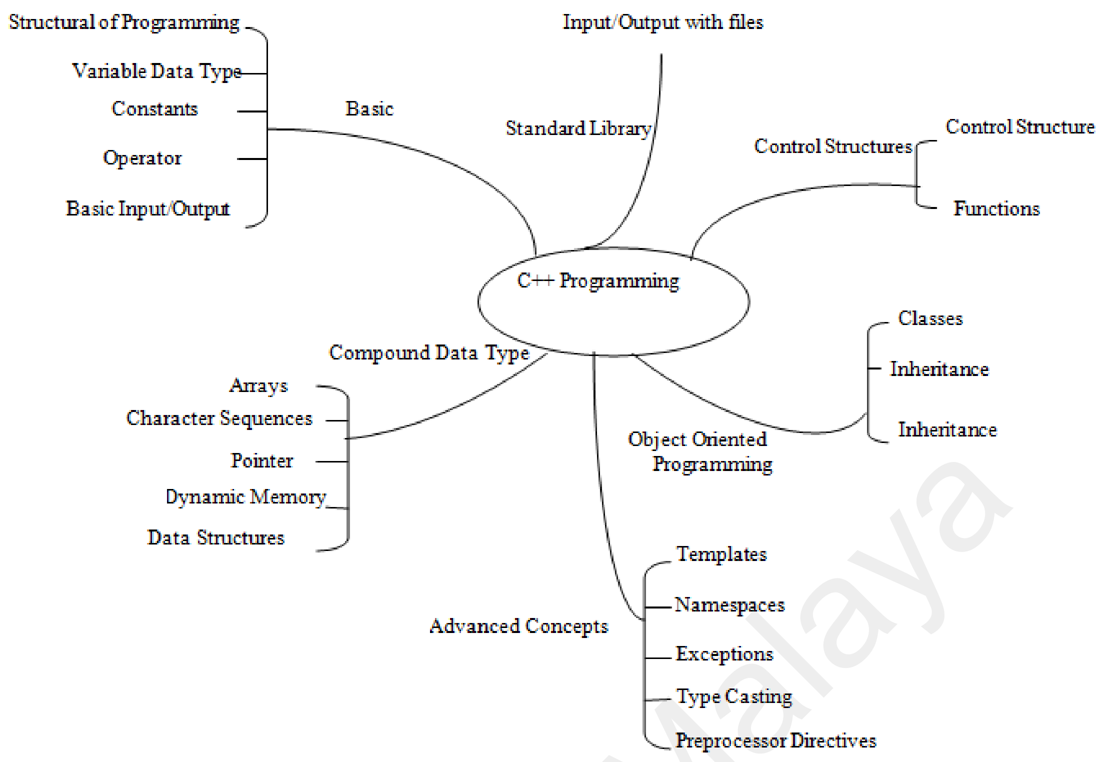


Figure 5.6: C++ Educational Ontology

The easiest technique for many ontology developers is using the combination approach top-down and bottom-up, since the "in the middle" concept tends to be the more descriptive concept in the domain (Rosch, 1999). Figure 5.6 depicts the sketch of C++ educational ontology that is composed of the main learning topic (e.g. basic, standard library, control structure) and sub-learning topic (e.g. arrays, character sequences, classes). The three different levels (top level, middle level and bottom level) of the MSSNP ontology are presented in Figure 5.7. The terms were extracted from (Deitel, Deitel, & Nieto, 1994).

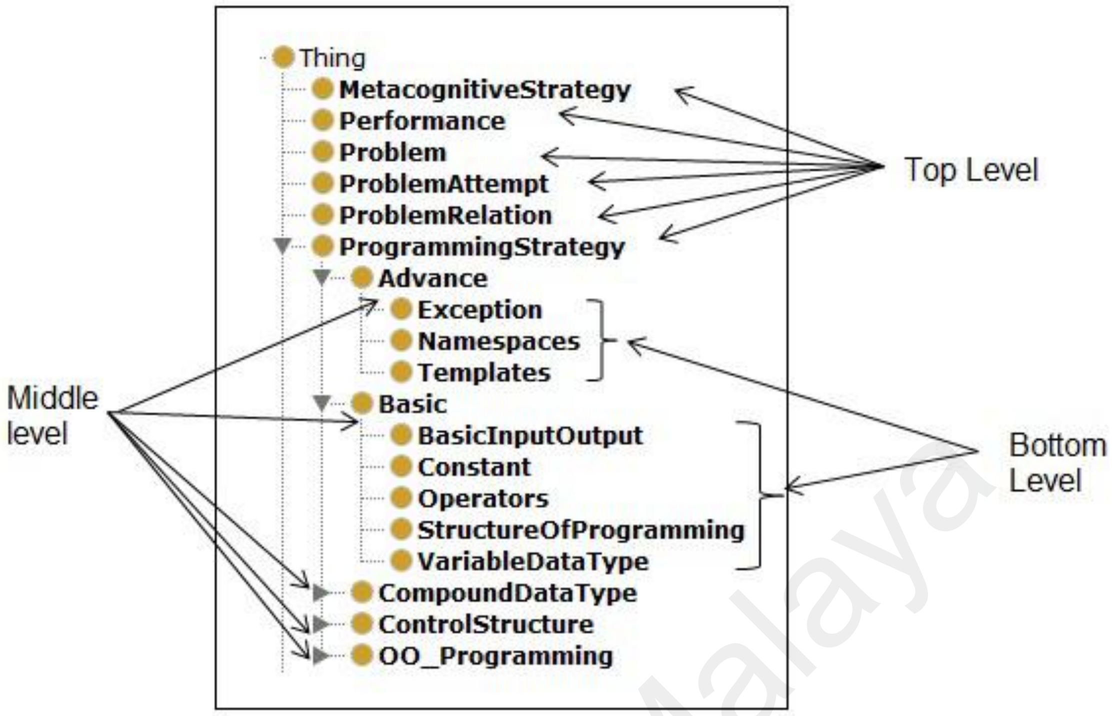


Figure 5.7: The three different hierarchies of the MSSNP Ontology

5.2.5 Property identification

In order to answer the competency questions, classes alone will not be sufficient to provide the information. Once the classes have been identified, the concept's internal structure must be described. In step 3, we have created classes selected from the list of terms. Most of the training terms are likely to be properties of these classes. These terms include, for example, a property in OWL describes the relationship among the classes. There are two main types of property: object and data type. The datatype properties link individuals to data value, while object properties are associated with the relationship between individuals. The third property is the annotation property, which can be utilized to add information.

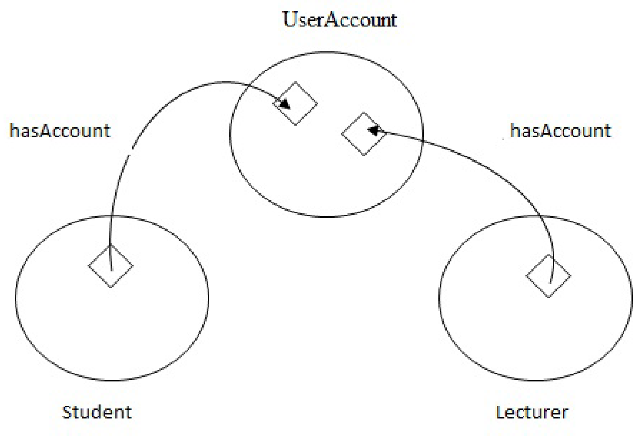


Figure 5.8: ‘hasAccount’ functional property

There are two types of object properties involved in the MSSNP ontology – the functional properties and inverse properties. As illustrated in Figure 5.8, functional properties are those that are limited to one unique relationship to another individual for a specified individual, i.e. there cannot be two distinct values y1 and y2 such that the pairs (x,y1) and (x,y2) are both instances of this property. In this study, the functional properties are applied in which a student and a lecturer can have one account to access the MSSNP application. In Inverse Functional Properties – Properties that describe the individual/domain are the inverse of another individual/domain. In the MSSNP ontology, the domain and the range for the hasSolution property and its inverse property is the isSolution. As shown in Figure 5.0, the domain of hasSolution is the problem and the range of hasSolution is ProblemSolution, the domain and range for isSolutionOf are the domain and range for hasSolution swapped over. Table 5.2 provides the list of all the object properties that the MSSNP comprises.

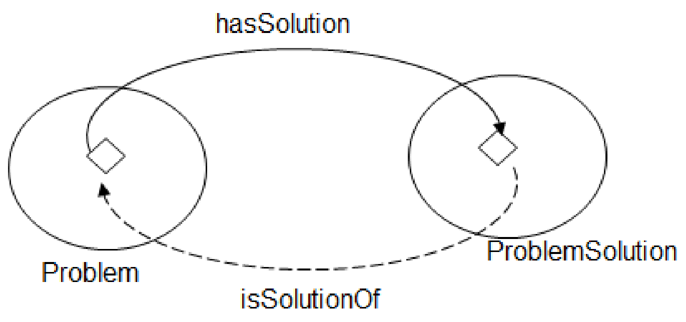
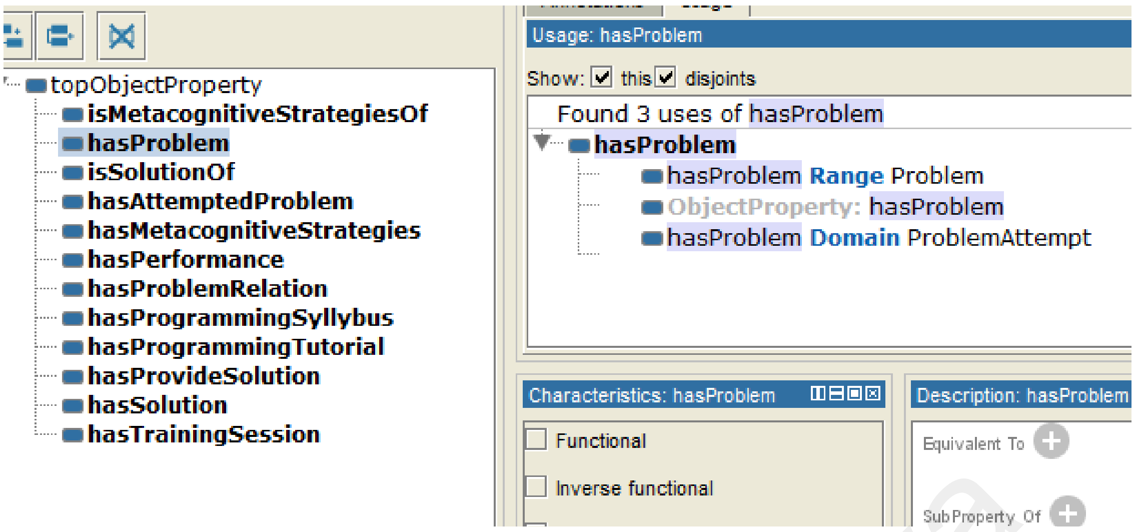


Figure 5.9: The domain and range for the ‘hasSolution’ property

Table 5.2: List of the MSSNP Object Property

Object Property	**Property Description**
hasAccount	Functional properties of domain Student and Lecturer with range UserAccount
hasAttemptedProblem	Properties of domain Student with range ProblemAttempt
hasKnowledgeComparison	Properties of domain Student and Lecturer with range UserAccount
hasLibraryProblem	Properties of domain Problem and Lecturer with range LibraryProblem
hasMultiAnswer	Properties of domain Student and Lecturer with range UserAccount
hasPerformance	Properties of domain Student with range Performance
hasProblem	Properties of domain ProblemAttempt with range Problem
hasProblemRelation	Properties of domain Problem with range ProblemRelation
hasProgrammingSyllybus	Properties of domain Problem with range ProblemStrategy
hasProgrammingTutorial	Properties of domain Problem with range ProblemStrategy
hasProvideSolution	Properties of domain Lecturer with range ProblemSolution
hasSolution	Inverse Properties of isSolution and domain Problem a with range ProblemSolution
hasStudentPerformance	Properties of domain Student range StudentPerformance
hasTrainingSession	Properties of domain Student with range TimeInterval
isMetacognitiveStrategiesof	Properties of domain MetacognitiveStrategy with range Problem
isSolutionof	Inverse Properties of hasSolution and domain ProblemSolution a with range Problem
hasKMA	Properties of domain Student with range StudentKMA

An object property is defined as an instance of the built-in OWL class owl:ObjectProperty. A datatype property is defined as an instance of the built-in OWL class owl:DatatypeProperty. Both owl:ObjectProperty and owl:DatatypeProperty are subclasses of the RDF class rdf:Property. Figure 5.10 shows part of the object properties created in the MSSNP Ontology.



topObjectProperty
- isMetacognitiveStrategiesOf
- hasProblem
- isSolutionOf
- hasAttemptedProblem
- hasMetacognitiveStrategies
- hasPerformance
- hasProblemRelation
- hasProgrammingSyllybus
- hasProgrammingTutorial
- hasProvideSolution
- hasSolution
- hasTrainingSession

Usage: hasProblem

Show: ☑ this ☑ disjoints

Found 3 uses of hasProblem

- hasProblem
  - hasProblem Range Problem
  - ObjectProperty: hasProblem
  - hasProblem Domain ProblemAttempt

Characteristics: hasProblem

☐ Functional

☐ Inverse functional

Description: hasProblem

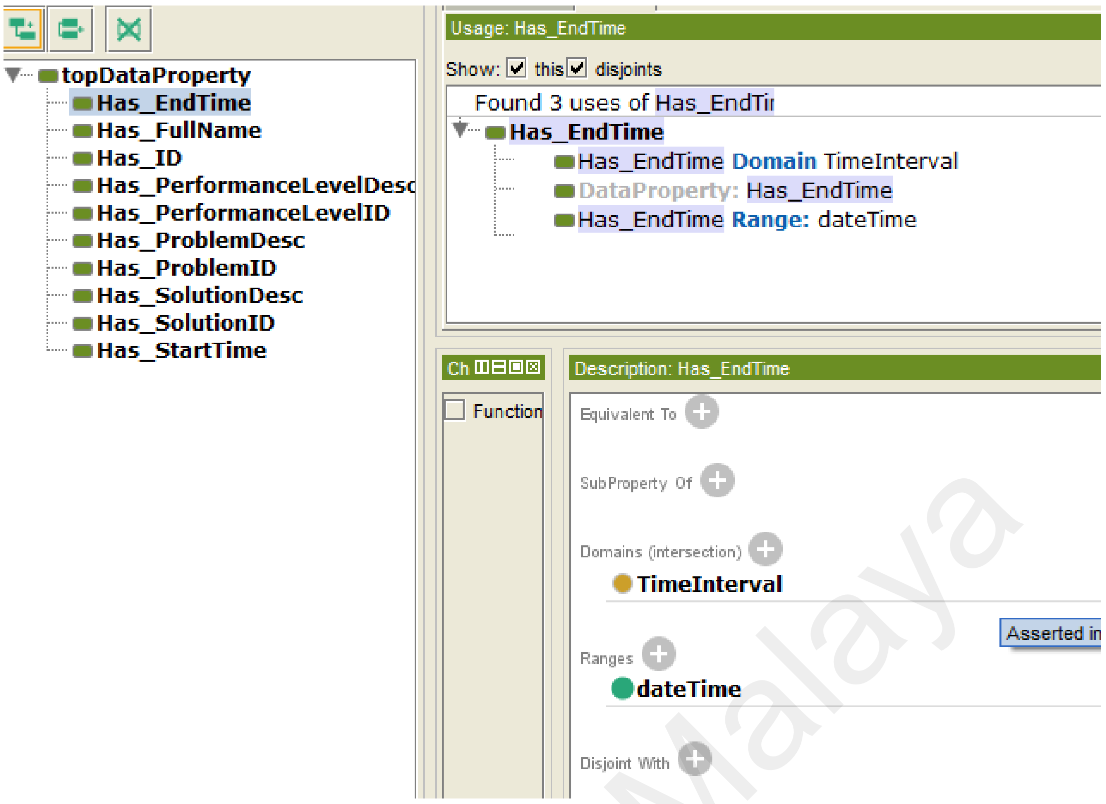
Equivalent To

SubProperty Of

Figure 5.10: Part of object properties in the MSSNP Ontology

5.2.6 Data property identification

Data properties connect the individual to rdf or XML schema datatype. They describe relationships between an individual and data value. OWL, the ontology structure of the MSSNP in this study has several data properties. OWL makes use of the RDF datatyping scheme, which provides a mechanism for referring to XML Schema datatypes (XML Schema Datatypes). Data values are instances of the RDF Schema class rdfs:Literal. Literals can be either plain (no datatype) or typed. Datatypes are instances of the class rdfs:Datatype. Figure 5.11 presents part of the MSSNP datatypes created using Protégé.



topDataProperty
Has_EndTime
Has_FullName
Has_ID
Has_PerformanceLevelDesc
Has_PerformanceLevelID
Has_ProblemDesc
Has_ProblemID
Has_SolutionDesc
Has_SolutionID
Has_StartTime

Usage: Has_EndTime
Show: ✓ this ✓ disjoints
Found 3 uses of Has_EndTi
Has_EndTime
Has_EndTime Domain TimeInterval
DataProperty: Has_EndTime
Has_EndTime Range: dateTime

Ch
☐ Function

Description: Has_EndTime
Equivalent To
SubProperty Of
Domains (intersection)
TimeInterval
Asserted in
Ranges
dateTime
Disjoint With

Figure 5.11: Data properties in the MSSNP

Table 5.3 presents part of the MSSNP datatypes that are used to describe and connecting each classes with data value.

Table 5.3: Part of the MSSNP Datatype

Data Property Name	**Data Property Description**
`Has_AnswerURL`	Properties of domain AnswerExplanation with range String
`Has_EndTime`	Properties of domain TimeInterval with range dateTime
`Has_FullName`	Properties of domain Student and Lecturer with range String
`Has_ID`	Properties of domain Lecturer, ProblemAttempt, Student, TimeInterval, LibraryProblem, StudentPerformance with range String
`Has_LibraryURL`	Properties of domain LibraryProblem with range String
`Has_Password`	Properties of domain UserAccount with range String
`Has_PerformanceLevelDecs`	Properties of domain Performance with range String
`Has_PerformanceLevelID`	Properties of domain Performance with range Int
`Has_Estimation`	Properties of domain ProblemAttempt with range Int
`Has_ProblemDesc`	Properties of domain Problem with range String
`Has_ProblemID`	Properties of domain ProblemMultipleAnswer, ProblemSolution, AnswerExplanation, Problem and LibraryProblem with range Int

`Has_SessionID`	Properties of domain TimeInterval,ProblemAttempt with range dateTime
`Has_SolutionDesc`	Properties of domain ProblemSolution with range String
`Has_SolutionID`	Properties of domain ProblemSolution,ProblemMultipleAnswer, ProblemAttempt with range int
`Has_TotalPrediction`	Properties of domain StudentPerformance with range int
`Has_TotalTimeInterval`	Properties of domain StudentPerformance with range int
`Has_Username`	Properties of domain UserAccount with range string

The Has_AnswerURL is a data property that is used to describe a domain named AnswerExplanation class with the string type of data range. The Has_ID

```
<owl:DatatypeProperty
rdf:about="&Ontology1373265476454;Has_SyllabusDesc">
<rdfs:domain rdf:resource="&Ontology1373265476454;Advance"/>
<rdfs:domain rdf:resource="&Ontology1373265476454;Arrays"/>
<rdfs:domain rdf:resource="&Ontology1373265476454;Basic"/>
<rdfs:domain rdf:resource="&Ontology1373265476454;BasicInputOutput"/>
<rdfs:domain
rdf:resource="&Ontology1373265476454;CharacterSequences"/>
<rdfs:domain rdf:resource="&Ontology1373265476454;Classess"/>
<rdfs:domain rdf:resource="&Ontology1373265476454;CompoundDataType"/>
<rdfs:domain rdf:resource="&Ontology1373265476454;Constant"/>
<rdfs:domain rdf:resource="&Ontology1373265476454;ControlStructure"/>
<rdfs:domain rdf:resource="&Ontology1373265476454;DataStructure"/>
<rdfs:domain rdf:resource="&Ontology1373265476454;DynamicMemory"/>
<rdfs:domain rdf:resource="&Ontology1373265476454;Exception"/>
```

The script presented above is part of the data property in the MSSNP ontology. The rdfs:domain axiom in the script above is a rdf built-in property that is used to link a property to a class description. The rdfs:range is used in this study to either link the property to the class description or a data range, whereas the rdfs:inverseof is used to define the relation between two classes.

5.2.7 Anomaly validity check

Of the many applications that are developed using Semantic Web technology, reasoning is one of the crucially important features. Descriptive Logics provide effective algorithms for reasoning for handling the OWL DL fragment. DL reasoner existed long before OWL. DL lacks some of the features required by Semantic Web applications (e.g. individuals reasoning, nominal support and querying capabilities). With these objectives, (Horrocks & Sattler, 2001) developed an expressive descriptive logic called Pellet, which is based on tableaux algorithms that support the OWL DL constructs that include owl:hasValue and owl:oneOf. Pellet provides complete algorithms reasoning for OWL DL. Figure 5.12 shows the main components of the Pellet reasoner.

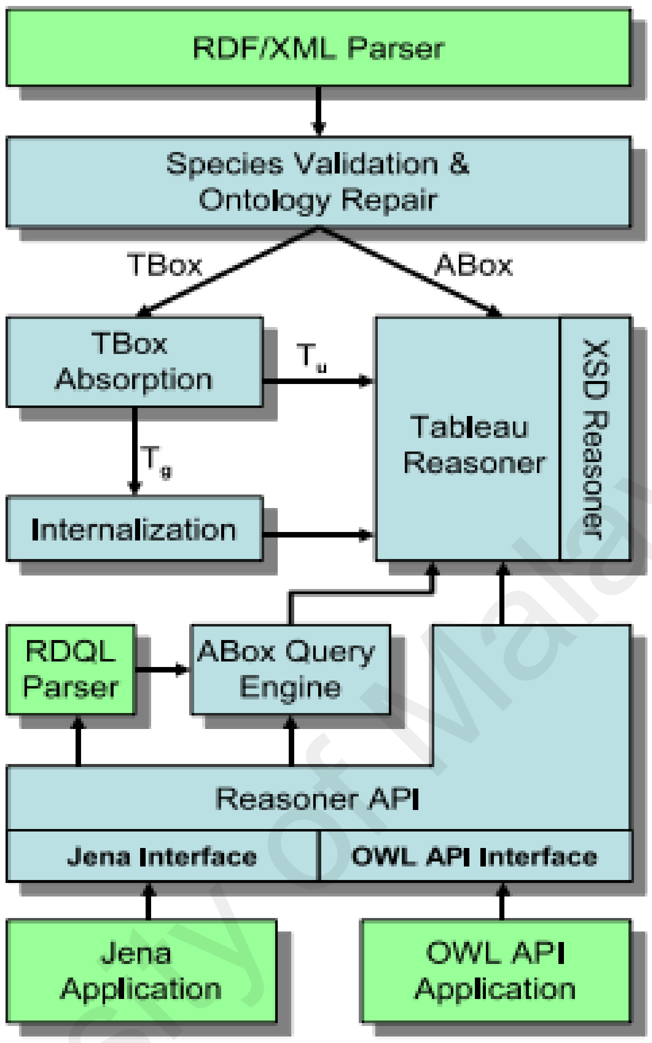


Figure 5.12: Pellet reasoned architecture

Pellet has the capability to do validation while the triples of the ontology are converted to axioms and assertions in the knowledge base. Pellet can also provide the correction capability for the ontology if there is missing type of triple by using some heuristics. It stores the classes' axiom in the TBox component and keeps the individual assertions in the ABox component. Pellet is available under the license of MIT and was implemented in pure Java. The anomalies check is the final step of the construction process to check the inconsistency of the ontology design. Using a reasoner Pellet, the process took 0.756 seconds to check the consistency concept. The overall MSSNP ontology is illustrated in Figure 10 using OWLViz. Figure 5.13 is the final look at the MSSNP Ontology.

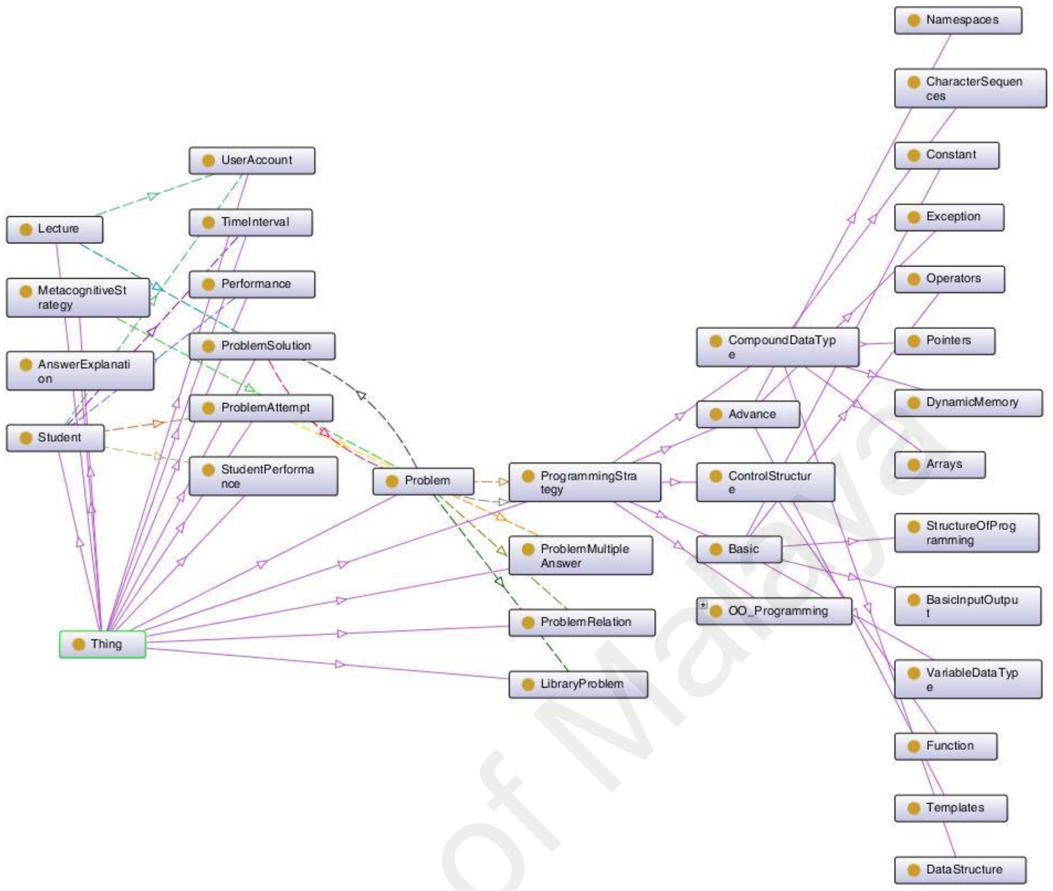


Figure 5.13: The final look at the MSSNP Ontology

5.3 Segment B- Triplestore data or Semantic Storage

A triplestore is a purpose-built database for the storage and retrieval of triples, a triple being a data entity composed of subject-predicate-object, like "Bob is 35" or "Bob knows Fred". Much like a relational database, one stores information in a triplestore and retrieves it via a query language. Unlike a relational database, a triplestore is optimized for the storage and retrieval of triples. In addition to queries, triples can usually be imported/exported using Resource Description Framework (RDF) and other formats. The next section will further discuss the steps as well as the technology for the development of semantic storage for the MSSNP's system.

5.3.1 ARC2 Framework

The ARC2 library available in PHP was used for parsing through the OWL file created in the previous module. ARC2 is a flexible RDF system for Semantic Web and PHP. It provides SPARQL and easy RDF parsing for LAMP systems. The ARC2 library was made available in the PHP module, which was then used to query the ontology using SPARQL. ARC uses object-oriented code for its components and methods, but the processed data structures consist of simple associative arrays, which lead to faster operations and less memory consumption. Apart from a few special formats returned by the SPARQL engine (e.g. from SELECT or INSERT queries), ARC is built around two core structures: triple sets and resource indexes. A single triple array contains the following keys:

- **s** the subject value (a URI, Bnode ID, or Variable)
- **p** the property URI (or a Variable)
- the object value (a URI, Bnode ID, Literal, or Variable)
- **s_type** "uri", "bnode", or "var"
- **o_type** "uri", "bnode", "literal", or "var"
- **o_datatype** a datatype URI
- **o_lang** a language identifier, e.g. ("en-us")

Variables are generated by ARC's Turtle parser, which was implemented for the SPARQL processor and therefore extends the Turtle spec with support for features, such as single quotes around literals, and variables. Table 5.4 presents the file components of ARC2.

Table 5.4: ARC2 File components

Class	**Description**
ARC2_SPARQLScriptProcessor	ARC2 SPARQLScript Processor
ARC2_Store	ARC2 RDF Store
ARC2_StoreQueryHandler	ARC2 RDF Store Query Handler
ARC2_StoreDumper	ARC2 Store Dumper
ARC2_MemStore	ARC2 Memory Store
ARC2_RemoteStore	ARC2 Remote RDF Store
ARC2_StoreHelper	ARC2 base class
ARC2_TestHandler	ARC2 base class
ARC2_Reader	ARC2 Web Client
ARC2_RDFExtractor	ARC2 base class
ARC2_Resource	ARC2 Resource object
ARC2_LegacyXMLParser	ARC2 base class
ARC2_RDFParser	ARC2 RDF Parser (generic)
ARC2_LegacyJSONSerializer	ARC2 base class
ARC2_LegacyHTMLSerializer	ARC2 base class
ARC2_LegacyXMLSerializer	ARC2 base class
ARC2_RDFSerializer	ARC2 RDF Serializer

5.3.2 ARC2 Installation

The WAMPServer is used as a Windows web development environment. It allows Web applications to be developed with Apache2 and PHP, while a MySQL database and PHPMyAdmin allows it to easily manage a database. For Semantic Web code development, the installation of PHP libraries is required. The arc2-starter-pack was downloaded from https://github.com/tuukka/arc2-starter-pack. Figure 5 shows the directories and files in the arc2-starter-pack zip file.

5.3.3 MySQL Schema Installation

ARC2 is a PHP 5.3 library for working with RDF. It also provides a MySQL-based triple store with SPARQL support. ARC2 is a Semantic Web database layered over MySQL that facilitates the storage of extensible metadata imported or entered into a Scalar book. ARC2 uses MySQL for persistency and provides an endpoint for the

SPARQL query. The following is the script used to create the database to store the OWL.

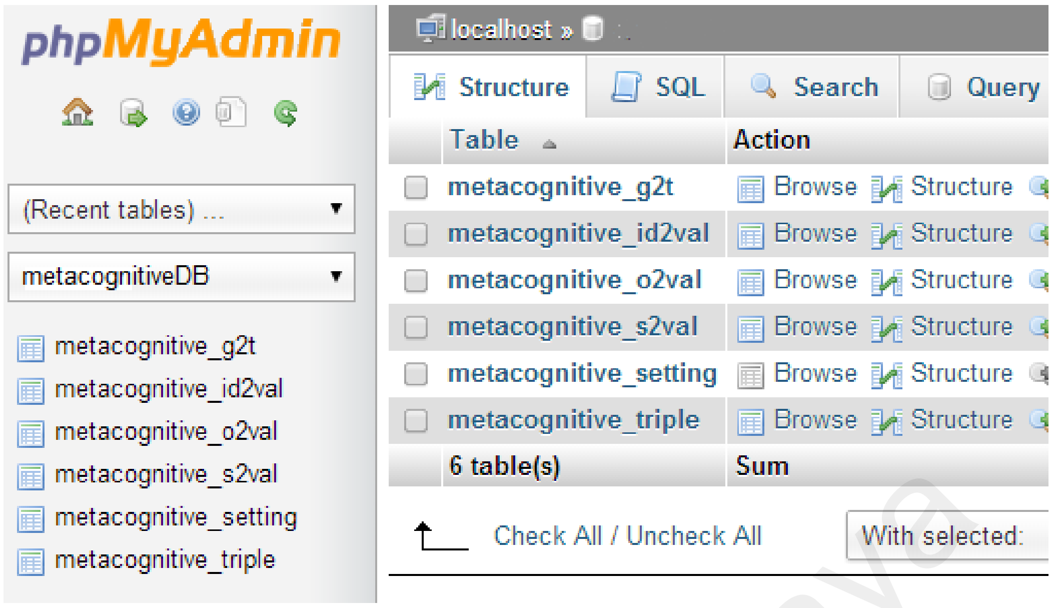
```
Create schema metacognitiveDB;
Grant all on metacognitiveDB.* to 'programmer_user'@'%' identified by
'********';
```

5.3.4 Loading OWL Data via PHP code

There are two ways to load OWL data into the RDF local store. The first is through the command line and the second is through the PHP application load. In this present work, we loaded the created MSSNP OWL via PHP application code. Below is the PHP script used to load the metacognitive.owl into MySQL.

```
  <?php
  include_once("/config/ARC2.php");
  $config = array(
   /* database info */
   'db_name' => 'metacognitiveDB',
   'db_user' => 'arc2test_user' ,
    'db_pwd' => '**********',
   /* store information */
   'store_name' => 'metacognitive',
   /* stop after 100 errors */
   'max_errors' => 100,
  );
  $store = ARC2::getStore($config);
  if (!$store->isSetUp()) {
   $store->setUp();
  }

  $parser = ARC2::getSemHTMLParser();
  $parser->parse('metacognitive.owl');
  $parser->extractRDF('rdfa');

  $triples = $parser->getTriples();
  $rdfxml = $parser->toRDFXML($triples);
?>
```

As the script above is executed, all tables to store triple-based data are created, as shown in Figure 5.14.Figure 5.15 presents the ARC SPARQL endpoint.

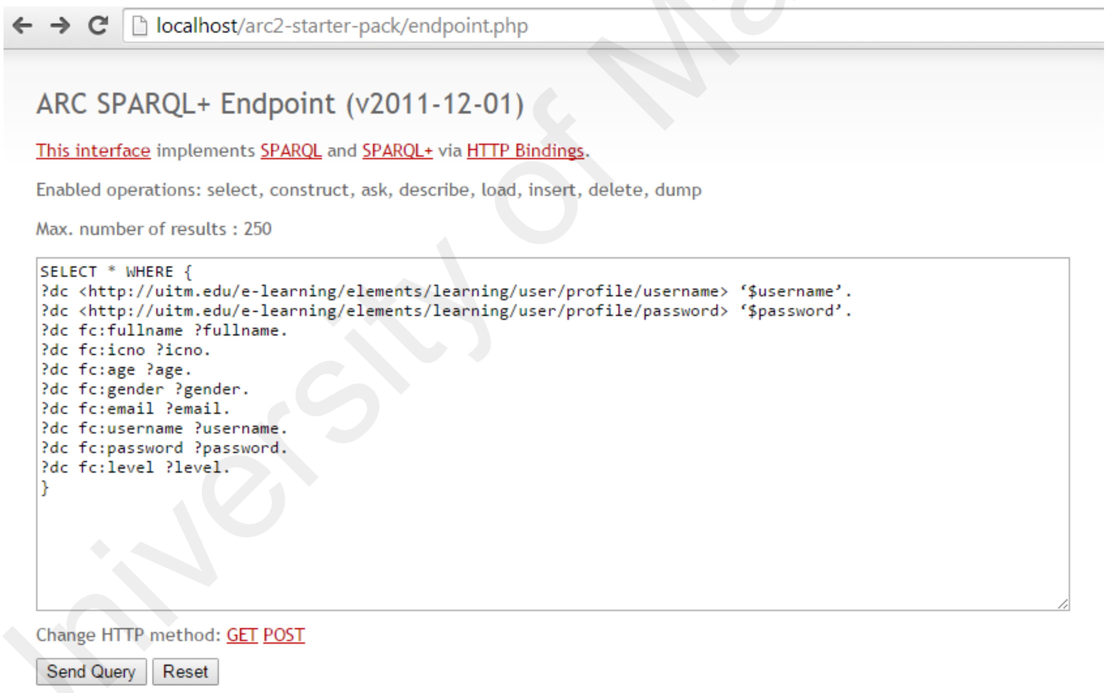


metacognitive_g2t
metacognitive_id2val
metacognitive_o2val
metacognitive_s2val
metacognitive_setting
metacognitive_triple

Table	Action
metacognitive_g2t	Browse Structure
metacognitive_id2val	Browse Structure
metacognitive_o2val	Browse Structure
metacognitive_s2val	Browse Structure
metacognitive_setting	Browse Structure
metacognitive_triple	Browse Structure
6 table(s)	Sum

Check All / Uncheck All   With selected:

Figure 5.14: Triple-based data store using MySQL



ARC SPARQL+ Endpoint (v2011-12-01)

This interface implements SPARQL and SPARQL+ via HTTP Bindings.

Enabled operations: select, construct, ask, describe, load, insert, delete, dump

Max. number of results : 250

```
SELECT * WHERE {
?dc <http://uitm.edu/e-learning/elements/learning/user/profile/username> '$username'.
?dc <http://uitm.edu/e-learning/elements/learning/user/profile/password> '$password'.
?dc fc:fullname ?fullname.
?dc fc:icno ?icno.
?dc fc:age ?age.
?dc fc:gender ?gender.
?dc fc:email ?email.
?dc fc:username ?username.
?dc fc:password ?password.
?dc fc:level ?level.
}
```

Change HTTP method: GET POST

Figure 5.15: ARC SPARQL endpoint

The script below is part of the data inserting process using a combination of PHP code and SPARQL script to register student information in order to allow the student to use the system.

```
$fullname = $_POST['fullname'];
$icno = $_POST['icno'];
```

```
  $age = $_POST['age'];
  $gender = $_POST['gender'];
  $email = $_POST['email'];
  $username = $_POST['username'];
  $password = $_POST['password'];
  $level = $_POST['level'];

  $store->query('LOAD <http://localhost/arc2-starter-
pack/metacognitive.owl>');
  if($store->query("
  PREFIX dc: <http://uitm.edu/E-learning/elements/learning/user/>
  PREFIX fc: <http://uitm.edu/E-
learning/elements/learning/user/profile/>
  INSERT INTO <metacognitive.owl>
  {
  dc:$icno fc:fullname '$fullname'.
  dc:$icno fc:icno '$icno'.
  dc:$icno fc:age '$age'.
  dc:$icno fc:gender '$gender'.
  dc:$icno fc:email '$email'.
  dc:$icno fc:username '$username'.
  dc:$icno fc:password '$password'.
  dc:$icno fc:level '$level'.
}"))
```

5.4 Segment C- Metacognitive Support Information for Novice Programmer

The main principles behind the MSSNP are to allow students to perform computer programming activities with minimal interference, following a metacognitive approach to learning. The MSSNP consists of three modules as discussed in the following sections.

5.4.1 Model for knowledge monitoring

The primary target of this framework is scaffolding the knowledge monitoring skill, in other words, to give the ability to assess one's knowledge, or, by extension, one understands. Promoting awareness of the novice's level in knowledge monitoring accuracy is the first step to foster metacognitive skill. Improvements in knowledge monitoring give the ability, which, in turn, triggers the selection attention and facilitates better allocation of cognitive resources. For measuring the knowledge (we called it KMA), we adapted the empirically validated instrument developed by Tobias &

Everson (Tobias & Everson, 1995) and made a minimum change to the empirical instrument by giving a little flexibility to the possibility that the student predicts that they would partially solve the problem or that they partially understood it. Table 5.5 presents the KMA with a score value for a, b, c, d, e, f, g, h, i. The KMA exhibits nine scores that reflect the relationship between a student's prediction of their knowledge and their actual performance. A score of 1 is given for the following situations, as follows:

a. Learner demonstrated not correct answer and estimated will not solve it (Condition a)
b. Learner demonstrated partially correct answer and estimated will solve partially (Condition e)
c. Learner demonstrated incorrect answer and estimated will solve (Condition i)

A score of -1 is given for the following situations:

a. Learner demonstrated incorrect answer and estimated will solve it (Condition c)
b. Learner demonstrated correct answer and estimated will not solve it (Condition g)

A score of -0.5 is given to the following situations:

a. Learner demonstrated not correct answer and estimated will solve it partially (Condition b)
b. Learner demonstrated partially correct answer and estimated will not solve it (Condition d)
c. Learner demonstrated partially correct answer and estimated will solve it (Condition f)
d. Learner demonstrated correct answer and estimated will solve it partially (Condition h) (Appendix K presents an example of flow for partially correct answer)

Table 5.5: KMA Score Condition

Prediction	**Unable to Solve It**	**Able to Solve It partially**	**Able to Solve It**
Actual Performance			
Provides Incorrect Answer	a	b	c
Provides partial correct answer	d	e	f
Provides Correct Answer	g	h	i

The mean of the KMA scores over all the problems solved, yields the current KMA state of the Student. Table 5.6 presents the classification of the KMA value, students with a KMA value between -1 and -0.25 are categorized as someone with a weakness for estimating correctly their knowledge level in the majority of situations. Whereas a student who is average in correctly estimating their knowledge level is indicated by a KMA value between -0.25 and 0.5. A high KMA (0.5 -1) shows that they are able to estimate their knowledge level correctly most of the time.

Table 5.6: Classification of KMA

KMA Value	**KMA Classification**	**Interpretation**
-1 to -0.25	Low	Weakness at estimating knowledge level correctly in majority of situations
-0.25 to 0.5	Average	Average in estimating knowledge level correctly, but makes frequent slightly wrong or completely wrong estimation
0.5 to 1	High	Most of the time makes correct estimation

5.4.2 Model for the evaluation of learning

The objective of this model is to develop awareness of how novices behave during problem-solving, which resources to use, how long one spends on each task, and which decisions novices make in the process of solving programming problem.

5.4.3 Model for the Selection of metacognitive strategies

The focus here is the general metacognitive heuristics that are connected to learning programming and the task. The focus is on developing students' awareness of three kinds of metacognitive strategy: strategies for monitoring understanding, strategies for monitoring the problem-solving process and controlling errors, and strategies for revising. The activities that the MSSNP comprises are discussed in the following sections.

5.5 Overview of activities in the MSSNP with the layout screen

The elements of metacognitive learning instruction involve actively thinking about what the learner knows and does not know, and how learners can get better at knowing and applying what they know. These elements have been considered in designing the MSSNP system. Following the MSSNP conceptual stages (as shown in Figure 5.3), the MSSNP system consists of five main activities, namely pre-task, familiarization, production, evaluation and post-task. The interaction with the MSSNP follows a pre-determined sequence of activities whose emphasis is either on metacognitive training or on problem-solving skills. The flow of activities and sequence is shown in Figure 5.16. Each of the activities, represented by a rectangular box, is clearly signaled to the student. In fact, every transition from one activity to the next is accompanied by a noticeable change of the user interface in order to offer only those functionalities relevant to the activity at hand. The rest of this chapter is organized around these activities.

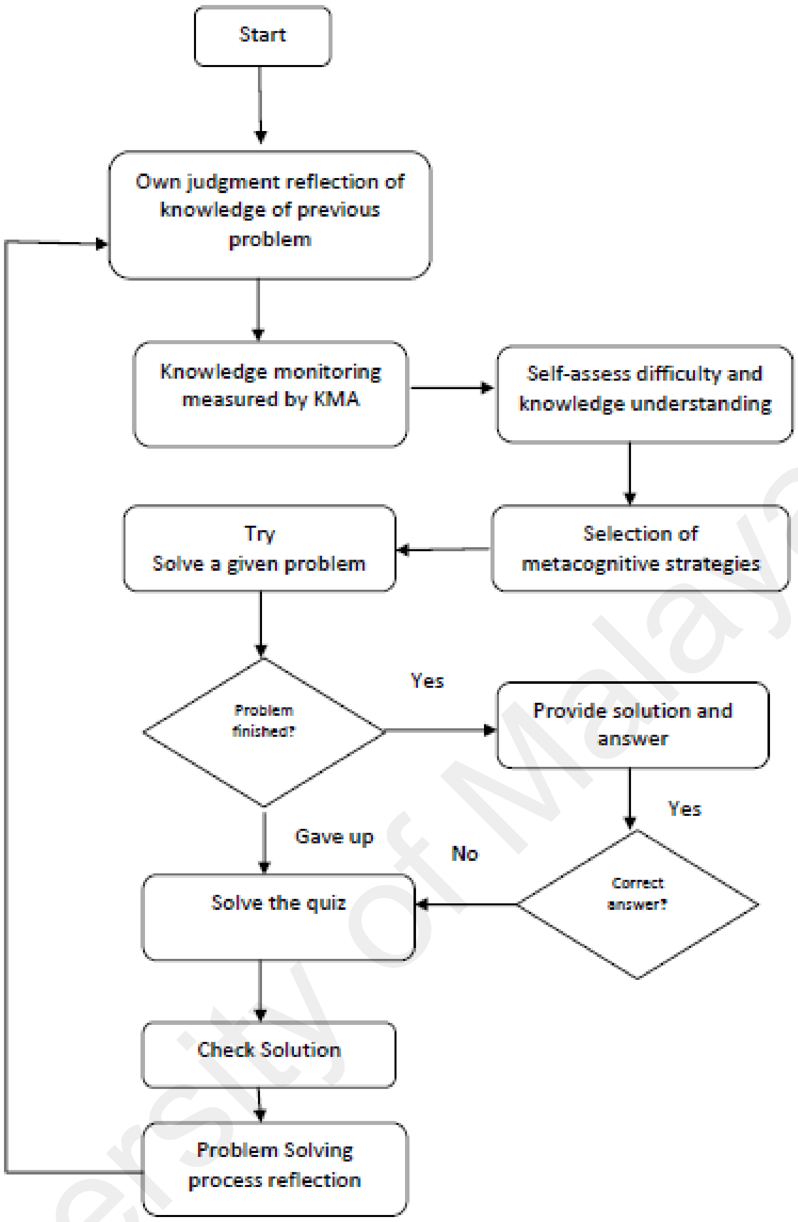
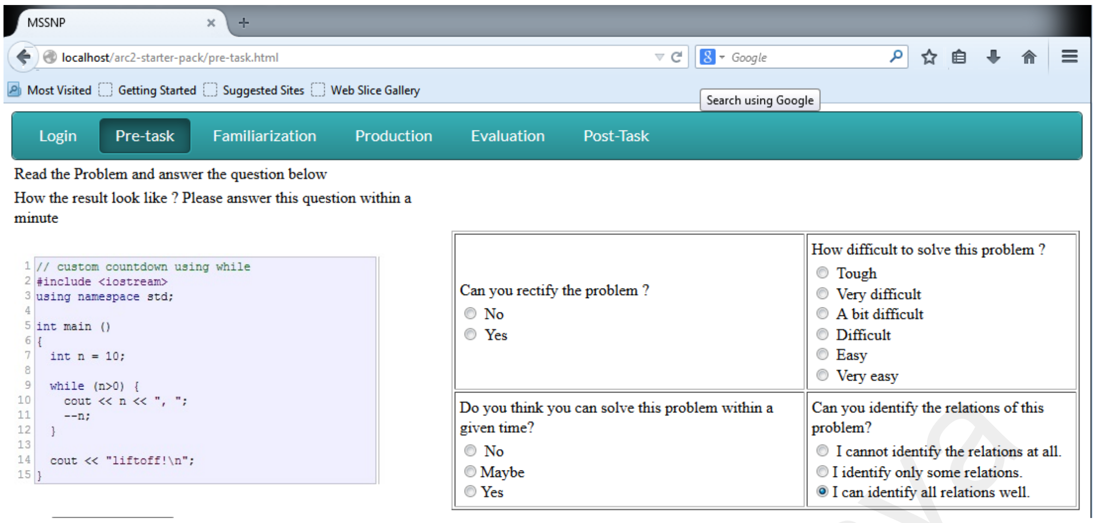

Start
Own judgment reflection of knowledge of previous problem
Knowledge monitoring measured by KMA
Self-assess difficulty and knowledge understanding
Selection of metacognitive strategies
Try
Solve a given problem
Problem finished?
Yes
Provide solution and answer
Yes
Correct answer?
No
Gave up
Solve the quiz
Check Solution
Problem Solving process reflection

Figure 5.16: The MSSNP activities sequence performed in one iteration

During the 'Own judgement reflection of Knowledge of previous problem', also called the 'Pre-Task' Stage, student is presented with the screen, as shown in Figure 5.17. As discussed in section 5.2.1 (Defining Scope and Boundary), the necessary metacognitive aspect is covered in this stage to start the new problem. This stage always concerns the self-reflection that happens before the learning process. Pre-task activity happens before the student embarks on the learning process. In this stage, the student is required to make comparisons between her judgment of her actual knowledge and her estimation of their knowledge. Figure 5.17 is the screen layout for pre-task activities.



Read the Problem and answer the question below

How the result look like ? Please answer this question within a minute

```
// custom countdown using while
#include <iostream>
using namespace std;

int main ()
{
  int n = 10;

  while (n>0) {
    cout << n << ", ";
    --n;
  }

  cout << "liftoff!\n";
}
```

Can you rectify the problem ? ○ No ○ Yes	How difficult to solve this problem ? ○ Tough ○ Very difficult ○ A bit difficult ○ Difficult ○ Easy ○ Very easy
Do you think you can solve this problem within a given time? ○ No ○ Maybe ○ Yes	Can you identify the relations of this problem? ○ I cannot identify the relations at all. ○ I identify only some relations. [✓ I can identify all relations well.]

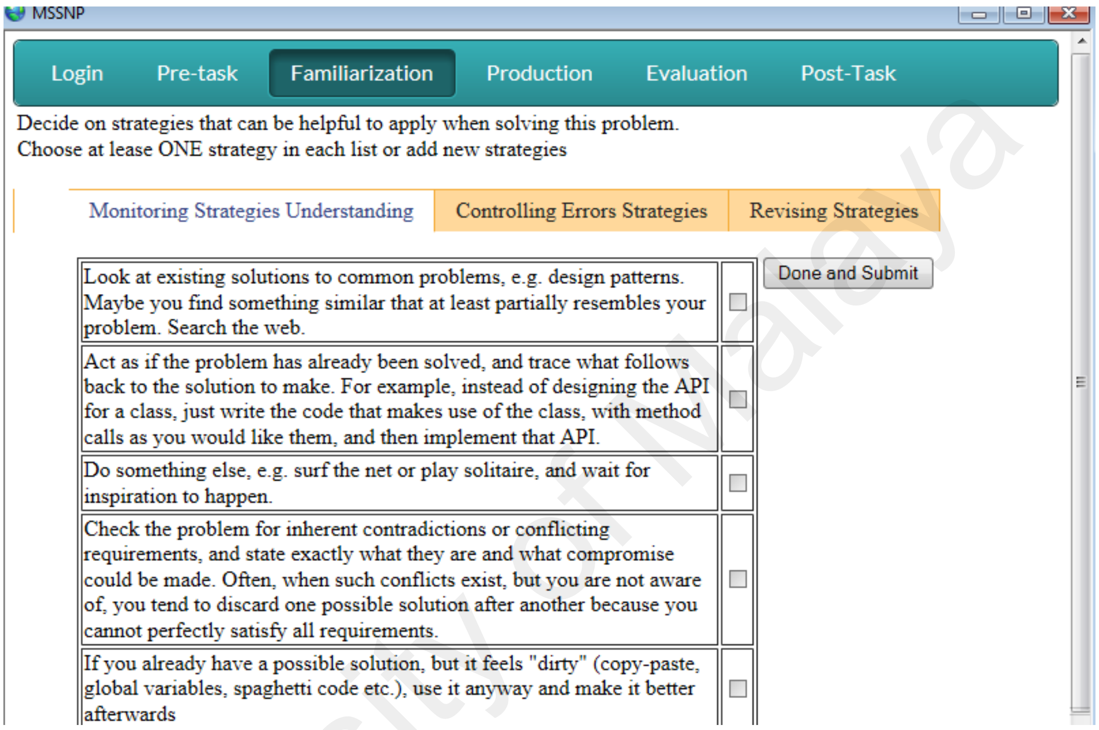
Figure 5.17: Reflect / Pre-Task Screenshot

Students are provided with suitable conditions for making them realize the benefits of general strategies, available resources as well as the degree of attention that is necessary to succeed in the problem-solving process and the activity. The objective of this stage is to trigger reflection on the student knowledge monitoring progress. It focuses on the past problems and performance (low, average or high) of students and compares their estimation and judgment to their actual knowledge understanding. In this stage, the student is exposed to the following activities:

- Knowledge monitoring and performance comparison
- Analysis of knowledge monitoring state, this is where the KMA is applied

```
<?php

  $rectify_prob = $_POST['rectify_prob'];
  $solve_prob = $_POST['solve_prob'];
  $prob_level = $_POST['prob_level'];
  $prob_relationship = $_POST['prob_relationship'];
  $icno = $_POST['icno'];

  $store->query('LOAD <http://localhost/arc2-starter-
pack/metacognitive.owl>');
  if($store->query("
  PREFIX dc: <http://uitm.edu/E-learning/elements/learning/user/>
  PREFIX km: <http://uitm.edu/E-
learning/elements/learning/user/KnowledgeMonitoring/>
  INSERT INTO <metacognitive.owl>
  {
  dc:$icno km:rectify_prob '$rectify_prob'.
  dc:$icno km:solve_prob '$solve_prob'.
  dc:$icno km:prob_level '$prob_level'.
  dc:$icno km:prob_relationship '$prob_relationship '.
  dc:$icno km:icno '$icno'.

}"))
```

Script above show the how the information in Pre-Task activities as showed in Figure 5.17 semantically stored in semantic storage.

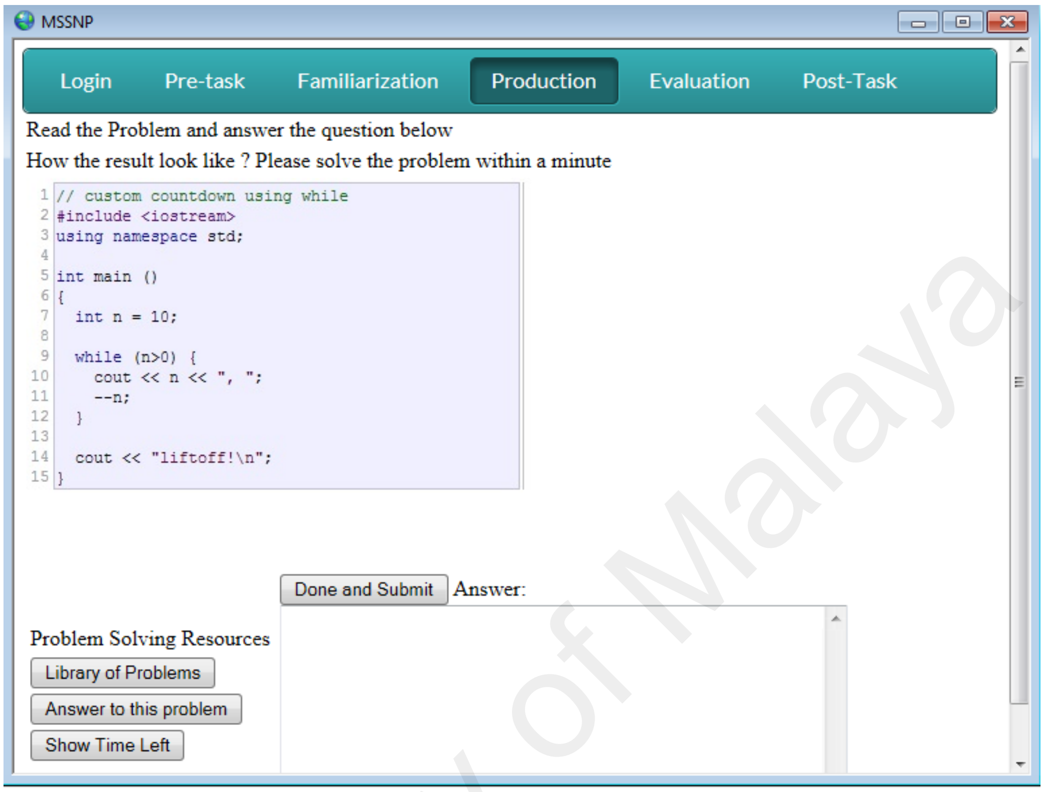


Decide on strategies that can be helpful to apply when solving this problem.
Choose at lease ONE strategy in each list or add new strategies

Look at existing solutions to common problems, e.g. design patterns. Maybe you find something similar that at least partially resembles your problem. Search the web.	☐
Act as if the problem has already been solved, and trace what follows back to the solution to make. For example, instead of designing the API for a class, just write the code that makes use of the class, with method calls as you would like them, and then implement that API.	☐
Do something else, e.g. surf the net or play solitaire, and wait for inspiration to happen.	☐
Check the problem for inherent contradictions or conflicting requirements, and state exactly what they are and what compromise could be made. Often, when such conflicts exist, but you are not aware of, you tend to discard one possible solution after another because you cannot perfectly satisfy all requirements.	☐
If you already have a possible solution, but it feels "dirty" (copy-paste, global variables, spaghetti code etc.), use it anyway and make it better afterwards	☐

Figure 5.18: Familiarization screenshot

Referring to the activity flowchart in Figure 5.16, the next activity concerns 'Knowledge Monitoring', which refers to the 'Familiarization' stage in the MSSNP system. During this task, a list of possible strategies is presented to the students who are required to select the appropriate strategies in order to solve a given problem. The strategies are divided into three types, specifically, monitoring strategies, understanding, controlling error strategies and revising strategies, as shown in Figure 5.18. The objective of this stage is for the student to reflect on the cognitive strategies that are available to solve the problem. The next activity is the 'self-assess difficulty and knowledge understanding', which is referred to as the 'Production' stage; the screenshot is presented in Figure 5.20. In this stage, the given task needs to be solved and answers need to be presented by the student. The objective of this stage is to reflect the student's understanding of the concept of programming as well as their confidence in solving the problem correctly. The student's performance takes place during this stage.



Read the Problem and answer the question below

How the result look like ? Please solve the problem within a minute

```
// custom countdown using while
#include <iostream>
using namespace std;

int main ()
{
  int n = 10;

  while (n>0) {
    cout << n << ", ";
    --n;
  }

  cout << "liftoff!\n";
}
```

Figure 5.19: Production screenshot

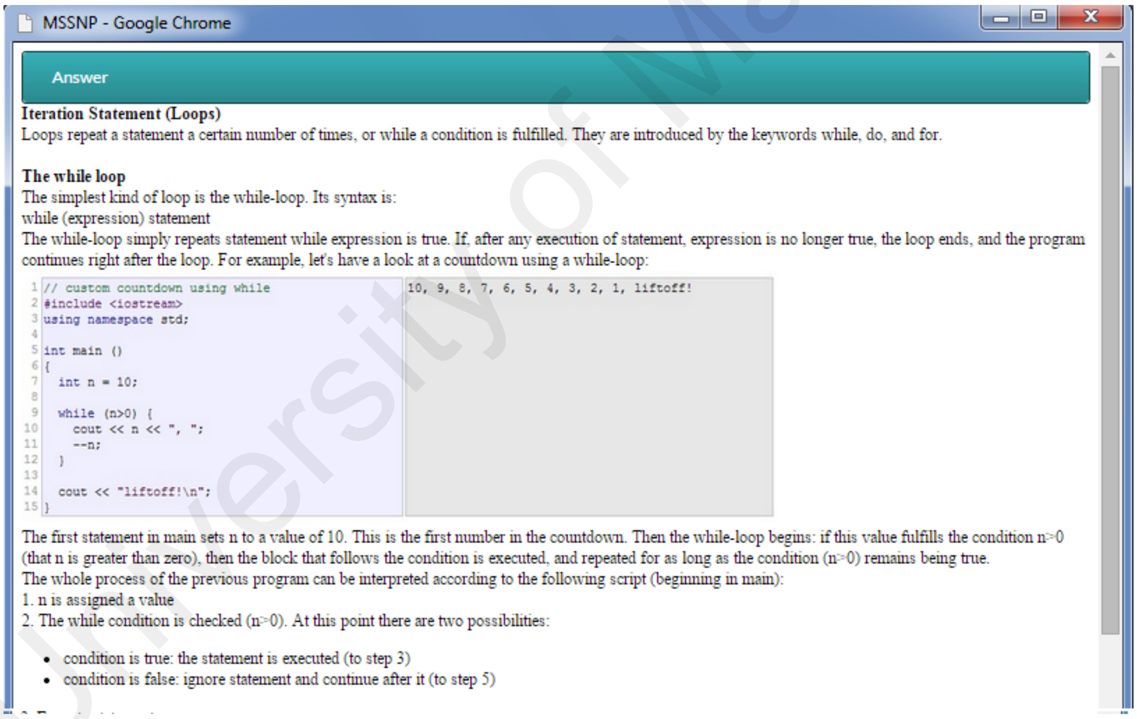
There are four buttons provided on the Production screen, as shown in Figure 5.19. The 'Done and Submit' button that allows the student to submit the answer and solution. The 'Library of Problems' presents all past problems solved. For each problem in the library, it shows the problem description, the student's answer, and the teacher's detailed solution. The 'Show Time Left' button displays or hides a countdown time. For each problem, the students have a maximum time to try the problem out. There are three distinct ways to reach the end of the problem attempt, as follows:

1. Gave up – student will choose this option if they want to give up at any point or if they think they cannot solve the problem in the time remaining. If they choose

this option, they will skip the "Checking Answer" screen and go straight to the "Quiz" screen.

2. Time is up – is an automatic option that appears if the student reaches the maximum time. A prompt window appears and the button 'Answer to this problem' is automatically enabled.
3. Finish – student successfully finishes the problem before the time is up, they then move to the Checking Answer activity.

The 'Answer to this problem' button is the solution for the given answer. By default the button is disabled, it is only enabled if the student fails to provide an answer within a given time. Figure 5.20 is the screen that displays the answer for the given question.



Iteration Statement (Loops)
Loops repeat a statement a certain number of times, or while a condition is fulfilled. They are introduced by the keywords while, do, and for.

The while loop
The simplest kind of loop is the while-loop. Its syntax is:
while (expression) statement
The while-loop simply repeats statement while expression is true. If, after any execution of statement, expression is no longer true, the loop ends, and the program continues right after the loop. For example, let's have a look at a countdown using a while-loop:

```
// custom countdown using while
#include <iostream>
using namespace std;

int main ()
{
  int n = 10;

  while (n>0) {
    cout << n << ", ";
    --n;
  }

  cout << "liftoff!\n";
}
```

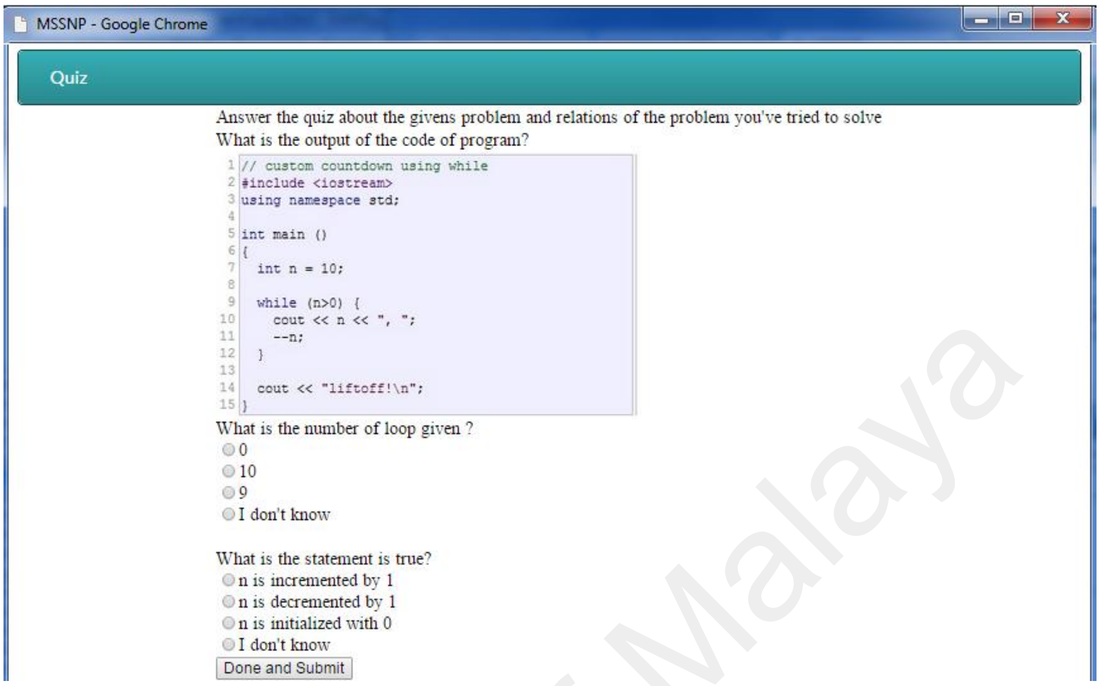
```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, liftoff!
```

The first statement in main sets n to a value of 10. This is the first number in the countdown. Then the while-loop begins: if this value fulfills the condition n>0 (that n is greater than zero), then the block that follows the condition is executed, and repeated for as long as the condition (n>0) remains being true.
The whole process of the previous program can be interpreted according to the following script (beginning in main):
1. n is assigned a value
2. The while condition is checked (n>0). At this point there are two possibilities:

- condition is true: the statement is executed (to step 3)
- condition is false: ignore statement and continue after it (to step 5)

Figure 5.20: Answer to the question

The student is presented with a quiz (e.g. as shown in Figure 5.21) whenever the answer chosen in the multiple choice (in the Check Answer activity) is either incorrect or correct, but the student acknowledges that it is different from their worked out answer. In 'Checking Answer' activity, whenever a student provides an answer to a problem they are asked to check if their answer is correct, and relate it to one of the answers provided in a multiple-choice question.



Answer the quiz about the givens problem and relations of the problem you've tried to solve
What is the output of the code of program?

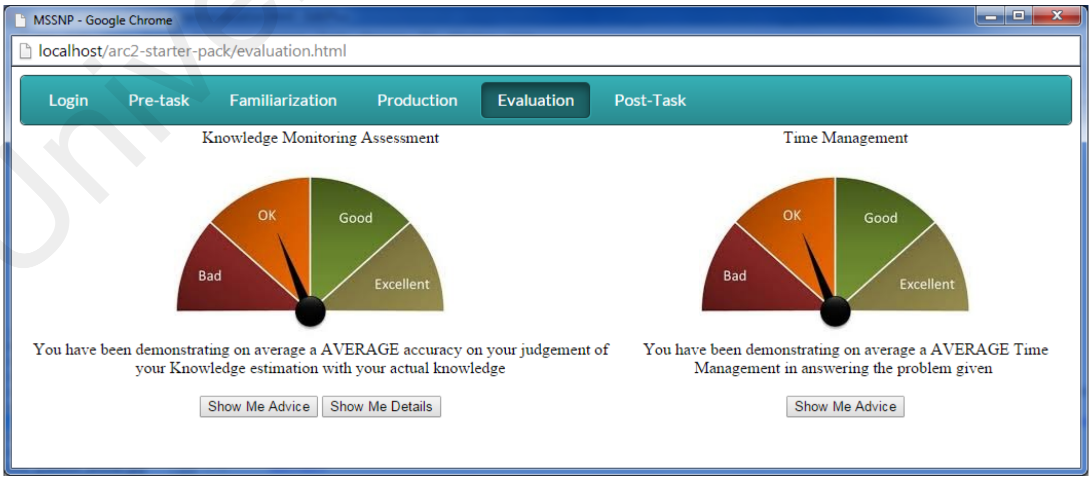
```
// custom countdown using while
#include <iostream>
using namespace std;

int main ()
{
  int n = 10;

  while (n>0) {
    cout << n << ", ";
    --n;
  }

  cout << "liftoff!\n";
}
```

What is the number of loop given ?
- 0
- 10
- 9
- I don't know

What is the statement is true?
- n is incremented by 1
- n is decremented by 1
- n is initialized with 0
- I don't know

Figure 5.21: Quiz Screenshot

There are two activities involved in 'Reflection of the past problem solution', that is, the 'Evaluation' (the screenshot of this activity is presented in Figure 5.22). The 'Evaluation' and 'Post-Task' activities take place after the problem is finished and the solution provided by the instructor is presented as the answer.

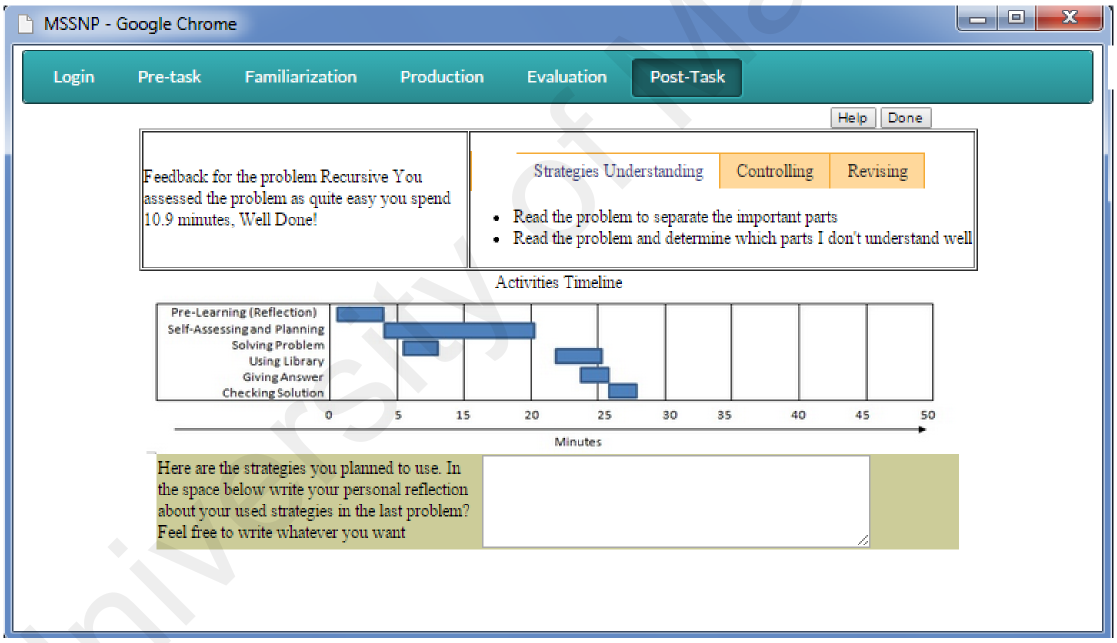


You have been demonstrating on average a AVERAGE accuracy on your judgement of your Knowledge estimation with your actual knowledge

You have been demonstrating on average a AVERAGE Time Management in answering the problem given

Figure 5.22: Evaluation Dashboard

The ‘Show Me advice’ button allows the student to see the advice that is auto-generated based on pre-defined variables (see Table 5.4). The student will be able to see the auto-generated advice concerning their knowledge monitoring assessment and time management during the problem solutions. In the ‘Post-Task’ activity, students are able to examine the activities in the Timeline Graph, read the feedback message and write a journal concerning the chosen and used strategies with the objective of reflecting their knowledge, as shown in Figure 5.23. We have designed graphical reification to convey the learner’s metacognition information in the form of reflectometers to trigger their knowledge monitoring accuracy and time management.



Feedback for the problem Recursive You assessed the problem as quite easy you spend 10.9 minutes. Well Done!

- Read the problem to separate the important parts
- Read the problem and determine which parts I don't understand well

Here are the strategies you planned to use. In the space below write your personal reflection about your used strategies in the last problem? Feel free to write whatever you want

Figure 5.23: Post-Task Screenshot

The final stage in the MSSNP is called the Post-stage where students will be given the opportunity to review their most recent experiences as well as explore things that happened during the activity of solving the problem. A timeline graph will be provided during this stage for the learner to be able to see the time spent on each task given. The content of the feedback message that is shown in the screen depends on a combination of factors based on the values of certain variables. The possible values and their meaning are presented in Table 5.7

Table 5.7: Pre-defined variables evaluated in the feedback message

Variable	Values	Description (Meaning)
Answer	Wrong	Student checked the wrong option in the multiple-choice.
	Gave up	No answer was provided because the student gave up on the problem.
	Partially-correct	Learner thinks worked-out answer is correct, but could not find equivalent equations from multiple choices, i.e. worked out solution had mistakes.
	Correct	Answer checked was similar to worked out solution
Check-Time	Not Low	Time checking the teacher's solution was greater or equal to 1.5 minutes
	Low	Time spent checking the teacher's solution was less than 1.5 minutes.
Difficulty	Low	Student rated the problem as "very easy" or "not difficult" prior to starting the problem.
	Average-High	Student rated the problem as "bit difficult", "very difficult" or "challenging".

5.6 Summary

Table 5.8 presents the comparison of features provided by the existing system with the MSSNP. This chapter has presented the principles and the design of the MSSNP, from the MSSNP ontological design, the semantic storage, and models to the application design. The existing work that has similar objectives for providing support tools for novice programmers in learning programming has been discussed in section 2.14.1, such as Gidget (Lee & Ko, 2011), CALMS (Thota & Whitfield, 2009), Jeliot (Moreno et al., 2004), JAVANIS (Oechsle & Schmitt, 2002) and ANNET (Liffick B.W. & Aiken R., 1996).

Table 5.8: Comparison of features between existing System and the MSSNP

System	Scaffolding	Modeling	Self-Assessment	Graphic Organizer	Self-Directed
Gidget (Lee & Ko, 2011)	√			√	√
CALMS (Liffick B.W. & Aiken R., 1996)	√	√			

Jeliot 3 (Moreno et al., 2004)	√	√		√	√
JAVAVIS (Oechsle & Schmitt, 2002)	√			√	
ANNET (Liffick B.W. & Aiken R., 1996)	√	√			√
MSSNP	√	√	√	√	√

However, these systems focus more on cognitive activities and elements. It is valuable if the instructional system could incorporate the metacognitive activities that focus on both the cognitive and social aspects of student development, including learning strategies and creation of supportive social environment for teaching and learning, as suggested by (Lin, 2001). As discussed in section 2.13, there are a few teaching techniques and self-directed strategies that have been commonly applied in classrooms that can be deliberately incorporated in the instructional learning environment, such as reflective prompts and questions, self-questioning, self-assessment and graphic organizers, as a pictorial way to organize information. To measure the effectiveness of the MSSNP, an experimental study is conducted with the aim being to investigate the effect of the interacting reflective activities introduced in the MSSNP as a tool for improving knowledge monitoring accuracy. A usability test using Software Usability Measurement Inventory (SUMI) is also conducted with the objective to identify the user's perception of MSSNP in terms of the affect, efficiency, control, helpfulness and learnability. The implementation and the result for both studies (experimental testing and usability testing) are discussed further in the next chapter.

CHAPTER 6: EXPERIMENTAL STUDY AND EVALUATION STUDY OF THE MSSNP

This chapter provides the evaluation of the metacognitive support environment for novice programmers, which has been discussed in the previous chapters. The first part of this chapter presents the experimental studies and puts forward the experimental hypotheses, together with the organization of the experimental. The evaluation specifically focuses on the usability test with the objective being to evaluate the affect, efficiency, control, helpfulness and the learnability of the MSSNP.

6.1 The effect of the MSSNP

In the previous chapter, we discussed the difficulties in designing the metacognitive support system environment for novice programmers, as it is hard to measure the metacognitive state and therefore establish whether the prototype system is effective and can achieve the objective. The MSSNP has been developed with the rationale that the metacognitive support environment has a positive impact on their learning gains. With metacognitive awareness, the student is encouraged to acknowledge their ability in knowledge monitoring, to choose appropriate metacognitive strategies as well as to evaluate their learning experience to improve the awareness of the student in the problem-solving process.

6.1.1 Experimental Design

The experimental design had a pre- and post-test and the group of subjects was divided into two, each associated with a different conditions. Similar to the idea for evaluation of Tobias and Everson (Tobias S. et al., 1999), the pre- and post-test were designed to measure the basic knowledge concepts in programming and knowledge monitoring. Table 6.1 shows the two groups of subjects each with specific conditions.

Table 6.1: The groups of subject and conditions

Group	**Conditions**
Experimental	Four weeks training on Introductory Computer Programming and interacted with the MSSNP to perform metacognitive activities throughout the training session
Control	Four weeks training on Introductory Computer Programming without interaction with the MSSNP

These tests (pre- and post-test) were divided into two sections: the first section (see Appendix G for Pre-Test Question Part 1 and Appendix I for Pre-Test Question Part 1) and the second section test (see Appendix H for Pre-Test Part 2 and Appendix J for Post-Test Part 2). In the first section, 3 minutes was given to the students to complete the section in which they were asked to estimate their knowledge in terms of whether or not they would be able to solve the problem given. For each problem, they had to answer "yes" or "no" for the question: "Do you think you can translate the problem given into pseudocode?" The participants were immediately redirected to the second section after completing the first section. In the second section, they were required to translate the same problems into pseudocode. The participants were given 10 minutes for the pre-test and 15 minutes for the post-test to complete the second section. The post-test was designed to be more difficult than the pre-test, given that the subjects would have gained more practice in solving those kinds of problem during the training. Both tests (pre-test and post-test) had five questions in total and were devised with the aid of a computer programming instructor. The scoring system is given as follows: 2.0 points for a correct answer, 1.0 point for a partially correct answer, 0.5 points for an incomplete answer and 0 for an incorrect answer. In this study, we are interested in comparing between the experimental and the control group and not in absolute measures of performance improvement in either group separately.

6.1.2 Participants

The participants consisted of a total of 33 first year undergraduate students of Computer Science who were divided into two groups – experimental and control. From the experimental group, 33.33% were male 66.67 were female from a total of 15 subjects. Whereas of the total of 18 in the control group, 27.78% were male and 72.22% were female. Their ages ranged between 21 and 25 years. The selected respondents for the "Motivation for computer programming problem-solving" item in the demographic form were distributed as follows: the majority of 75.75% selected fairly motivated, 21.21% very motivated and 3.04 % chose extremely motivated. Thus, no one in this group selected the not at all motivated or a bit motivated options in the 5-point scale. Table 6.2 presents the distribution of respondents by group and gender.

Table 6.2: Distribution of respondents by group

Group	**Frequency**			**Percentage %**		
	Male	**Female**	**Total**	**Male (%)**	**Female (%)**	**Total (%)**
Experimental group	5	10	15	33.33	66.67	100
Control group	5	13	18	27.78	72.22	100
Total	33			100		

For familiarity with computer assisted learning tools, 36.36% categorized themselves as having average familiarity, 33.33% stated unfamiliar, 12.12% identified themselves as newcomers and the other 13.52 as beginners.

6.1.3 Materials

Appendix N provides a list of all materials used for the experiment.

6.1.4 Preparation for data analysis

An exploratory examination of the data with inferential analysis was conducted with the use of SPSS Software.

Table 6.3: The results of KMA Pre-test, Pre-Test score, KMA Post-Test, Absolute Difference, Relative Difference and KMA difference of Control group

Respondents	KMA-Pre-Test	Pre-Test Score	KMA-Post-Test	Post-Test-Score	Absolute Difference (Post-Test – Pre-Test	Relative-Difference (Diff-Score-Test/Pre-Test Score)	KMA-different (KMA Post-Test-KMA Pre-Test)
R1	0.6	7	0.6	7	0	0.00	0
R2	0	8	0	8	0	0.00	0
R3	-0.3	3	-0.3	3	0	0.00	0
R4	0.6	10	0.6	10	0	0.00	0
R5	0	8	0	8	0	0.00	0
R6	0.4	8	0.4	8	0	0.00	0
R7	-0.1	7	0	6	-1	-0.14	0.1
R8	0.7	5	0.3	4	-1	-0.20	-0.4
R9	0.7	9	0.3	9	0	0.00	-0.4
R10	0.6	7	0.6	7	0	0.00	0
R11	0.3	3	0.3	3	0	0.00	0
R12	0.4	6	0.4	6	0	0.00	0
R13	0.1	6	0.1	6	0	0.00	0
R14	0.3	6	0.3	6	0	0.00	0
R15	0.4	7	0.4	7	0	0.00	0
R16	0.4	6	0.4	6	0	0.00	0
R17	0.7	10	0.7	10	0	0.00	0
R18	0.3	6	0.3	6	0	0.00	0

Table 6.3 and Table 6.4 show the scores of the KMA Pre-Test, Pre-Test, KMA Post-Test, Post-test, Absolute difference between Post-test score and Pre-Test score, Relative Difference and KMA different score of subjects in the control group and experimental group. The value of the Score of Absolute Difference was derived from the subtraction of the Post-test and Pre-Test score. Eighteen subjects participated in the control group and were named as R1, R2, R3, R4…R 18, and 15 subjects participated in the experimental group named as R1, R2, R3… R15. The post-test scores were obtained

with the aid of a computer programming instructor using the scoring system discussed in the previous section. The Absolute Difference value refers to the different value of the post-test score and pre-test score (Gumbel, 2012). The zero value of absolute difference shows that there is no difference in the performance of subjects before and after the training, the positive value shows that there is an improvement of knowledge monitoring performance after the training, whereas a negative value shows that the performance of the student is better before the training is given. The relative difference is defined as the mean absolute difference divided by the Pre-test score.

Table 6.4: The results of the KMA Pre-test, Pre-Test score, KMA Post-Test, Absolute Difference, Relative Difference and KMA difference of the Experimental group

Respondent	KMA Pre-Test	Pre-Test Score	KMA Post-Test	Post-Test-Score	Absolute Difference (Post-Test –Pre-Test)	Relative-Difference (Diff-Score-Test/Pre-Test Score)	KMA-different (KMA Post-Test-KMA Pre-Test)
R1	-0.2	3	-0.6	1	-2	-0.67	-0.4
R2	0	8	0.3	8	0	0.00	0.3
R3	-0.3	3	-0.3	3	0	0.00	0
R4	0.6	10	0.7	10	0	0.00	0.1
R5	0	8	0.3	8	0	0.00	0.3
R6	0.4	8	1	10	2	0.25	0.6
R7	-0.1	7	-0.1	5	-2	-0.29	0
R8	-0.4	5	0.3	7	2	0.40	0.7
R9	0.1	4	0.1	4	0	0.00	0
R10	-0.2	6	-0.2	6	0	0.00	0
R11	0.3	3	0.7	5	2	0.67	0.4
R12	0.4	6	0.7	10	4	0.67	0.3
R13	0.1	6	0.7	9	3	0.50	0.6
R14	0.3	6	0.6	7	1	0.17	0.3
R15	0.1	6	0.4	6	0	0.00	0.3

6.1.5 Data Normality

In statistics, a normality test is used to determine if a dataset is well-modeled by a normal distribution and to compute how likely it is for a random variable underlying the dataset to be normally distributed (Gumbel, 2012). Normality tests that involved Kolmogorov-Smirnov and Shapiro-Wilk were computed using SPSS on the main variables (e.g. pre-test scores, post-test scores, KMA in pre-test, and KMA in post-test). These revealed that most of the main variables are normally distributed (see results in Table 6.5 and Table 6.6), with the small sample size (experimental (N)=15 and control(N)=18),

Table 6.5: Normality Test for control group data

	Kolmogorov-Smirnov[a]			**Shapiro-Wilk**		
	Statistic	**df**	**Sig.**	**Statistic**	**df**	**Sig.**
KMAPreTestExp	.130	15	.200*	.973	15	.895
PreTestScoreExp	.179	15	.200*	.930	15	.270
KMAPostTest	.161	15	.200*	.945	15	.456
PostTestScore	.103	15	.200*	.950	15	.519
diffscoretestExp	.255	15	.010	.908	15	.128
relativedifferenceExp	.240	15	.020	.907	15	.121
KMAdifferentExp	.191	15	.146	.936	15	.332

*. This is a lower bound of the true significance.
a. Lilliefors Significance Correction

Table 6.5 presents the results of the normality test for the control group data, for the Kolmogorov-Smirnov and Shapiro-Wilk Test. The Shapiro-Wilk Test is more appropriate for small sample sizes (<50 samples). For this reason, we take the results of the Shapiro-Wilk test as our numerical means of assessing normality. From the table we can see that all variables (KMAPreTestExp, PreTestScoreExp, KMAPostTest, diffscoretestExp relativedifferenceExp and KMAdifferentExp) have a sig. value greater than 0.05, which indicates that the data are normally distributed. Table 6.6 presents the results of the normality test for the experimental group. The results show that most variables are normally distributed except absolutedifference, relativedifference and KMAdifferent. For further investigation, the paired sample t-tests were performed to investigate the group samples and the repeated measures differences, respectively to understand whether there was difference in student before and after 4 weeks training on Introductory Computer Programming and interaction with MSSNP.

Table 6.6: Normality Test for experimental group

Tests of Normality						
	Kolmogorov-Smirnov[a]			**Shapiro-Wilk**		
	Statistic	**df**	**Sig.**	**Statistic**	**df**	**Sig.**
KMAPreTest	.170	18	.180	.923	18	.145
PreTestScore	.179	18	.133	.938	18	.267
KMAPostTest	.222	18	.019	.933	18	.218
PostTestScore	.205	18	.045	.934	18	.228
absolutedifference	.523	18	.000	.373	18	.000
relativedifference	.521	18	.000	.386	18	.000
KMAdifferent	.504	18	.000	.463	18	.000
a. Lilliefors Significance Correction						

Type II errors can indicate a failure to reject a false null hypothesis, or, in other words, fail to detect an effect that is present. In order to avoid the possibility of Type II errors occurring due to the use of these less sensitive non-parametric tests, the paired sample tests were performed to determine whether there is a significant difference between the average values of the KMA pre-test and the post-test for both groups – experimental and control group.

Table 6.7: Paired Samples Statistics for Experimental group

Paired Samples Statistics for Experimental group					
		Mean	**N**	**Std. Deviation**	**Std. Error Mean**
Pair 1	KMAPreTestExp	.073	15	.2865	.0740
	KMAPostTest	.307	15	.4511	.1165

In the paired sample statistics for the experimental group, as shown in Table 6.7, the mean for KMAPreTest is 0.073; the mean for KMAPostTest is 0.307; the standard deviation for the KMAPreTest is 0.2865 and for the KMAPostTest it is 0.4511.

Basically, on average, a small value of standard deviation in the statistical data is close to the mean of the dataset, while a larger deviation is further away from the mean. Standard deviation as a single number that can be difficult to interpret. Therefore, the paired samples statistical analysis was performed and presented in the following section.

Table 6.8: Paired Samples Test for Experimental group

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	KMAPreTestExp - KMAPostTest	-.2333	.2895	.0747	-.3937	-.0730	-3.122	14	.008

Table 6.8 contains information about the paired sample t-test for the experimental group. The paired sample t-test is performed to compare the means between the KMAPreTest and KMAPostTest, or in simple words, to compare the differences of the performance indicated by the KMA values before and after the training. The Significant value (0.008 < 0.05) in Table 6.8 shows that there is statistically significant difference between the KMAPreTest and KMAPostTest for the experimental group or in other words there is a difference in terms of student KMA before and after the training. In the paired Sample statistics for the control group, as shown in Table 6.9, the mean for the KMAPreTest is 0.339. The mean for the KMAPostTest is 0.3. The standard deviation for the KMAPreTest is 0.2865 and for the KMAPostTest it is 0.4511.

Table 6.9: Paired Samples Statistics for Control group

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	KMAPreTest	.339	18	.2973	.0701
	KMAPostTest	.300	18	.2590	.0610

The Significant value (0.233 >0.05) in Table 6.10 indicated that there is no statistically significant difference between the KMAPreTest and KMAPostTest for the control group, or, in other words, there is no difference in the student performance in KMA before and after the training.

Table 6.10: Paired Samples Test for the Control group

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	KMAPreTest - KMAPostTest	.0389	.1335	.0315	-.0275	.1053	1.236	17	.233

Table 6.11: Descriptive Statistics for KMA in pre- and post-test for the experimental group

		KMAPreTestExp	KMAPostTest
N	Valid	15	15
	Missing	0	0
Mean		.073	.307
Median		.100	.300
Mode		.1	.7
Std. Deviation		.2865	.4511
Minimum		-.4	-.6
Maximum		.6	1.0
Sum		1.1	4.6
Percentiles	25	-.200	-.100
	50	.100	.300
	75	.300	.700

The descriptive values of KMA in the pre- and post-test for both groups (experimental and control) are shown in Table 6.11 and Table 6.12. In the experimental group an important two times increase was observed in the upper quartile (KMA Pre-test 75*th* percentile=0.3 and Post-test 75*th* percentile=0.7) (See Table 6.11), suggesting an improvement in this metacognitive skill for this group.

Table 6.12: Descriptive Statistics for KMA in the pre- and post-test for the control group

		KMAPreTest	KMAPostTest
N	Valid	18	18
	Missing	0	0
Mean		.339	.300
Median		.400	.300
Mode		.4	.3
Std. Deviation		.2973	.2590
Minimum		-.3	-.3
Maximum		.7	.7
Sum		6.1	5.4
Percentiles	25	.075	.075
	40	.300	.300
	75	.600	.450

The descriptive values of KMA in the pre- and post-test for the control group are shown in Table 6.12. In the control group, a decrease in the upper quartile (KMA Pre-test 75*th* percentile=0.6 and Post-test 75*th* percentile=0.450) was observed. The Pre-Test and the Post-Test of KMA frequency for the control group are provided in Table 6.13 and Table 6.14.

Table 6.13: Pre-Test KMA frequency for the control group

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-.3	1	5.6	5.6	5.6
	-.1	1	5.6	5.6	11.1
	.0	2	11.1	11.1	22.2
	.1	1	5.6	5.6	27.8
	.3	3	16.7	16.7	44.4
	.4	4	22.2	22.2	66.7
	.6	3	16.7	16.7	83.3
	.7	3	16.7	16.7	100.0
	Total	18	100.0	100.0	

Table 6.14: Post-Test KMA frequency for the control group

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	-.3	1	5.6	5.6	5.6
	.0	3	16.7	16.7	22.2
	.1	1	5.6	5.6	27.8
	.3	5	27.8	27.8	55.6
	.4	4	22.2	22.2	77.8
	.6	3	16.7	16.7	94.4
	.7	1	5.6	5.6	100.0
	Total	18	100.0	100.0	

6.1.6 Correlation Observation

In order to find the predictor factor or infer causality, a range of correlations were computed. The results are presented in a matrix (see Table 6.15 and Table 6.16) such that, as can be seen, the correlations are replicated. Nevertheless, the table presents Spearman's correlation, its significance value and the sample size that the calculation was based on:

- Correlation between the scores for the post-test and pre-test (see Table 6.15 and Table 6.16): we intend to investigate whether the pre-test scores were a good predictor of the post-test scores. The computed result show (Spearman's correlation coefficient, r_s, for experimental group is 0.753 with bilateral p-value=0.001 N=15 and Spearman's correlation coefficient, r_s, for control group is 0.981 experimental group with bilateral p-value=0.00 and N=18).
- Correlation between scores of Pre-Test and Pre-test KMA (see Table 6.15 and Table 6.16): (Tobias and Everson, 2002) indicates that the level of KMA is a good predictor for measuring performance. From the computed result, we found a positive correlation for both groups (Spearman's correlation coefficient, r_s, for the experimental group is 0.412 with bilateral p-value=0128 and N= 15. Spearman's correlation coefficient, r_s, for the control group is 0.348 experimental group with bilateral p-value=0.158 and N=18).

- Correlation between the post-test score and post-test KMA (seeTable 6.15 and Table 6.16).A closer look at the results showed that a positive correlation exists for both the experimental and the control group (Spearman's correlation coefficient, r_s, for the experimental group is 0.779 with bilateral p-value=0.001 and N= 15. Spearman's correlation coefficient, r_s, for the control group is 0.438 experimental group with bilateral p-value=0.069 and N=18).

Table 6.15: Positive Correlation Test with Spearman for Pre-test vs. Post-test scores of Experimental group

Correlation						
			KMAPreTestExp	**PreTestScoreExp**	**KMAPostTest**	**PostTestScore**
Spearman's rho	KMAPreTest	Correlation Coefficient	1.000	.412	.845**	.611*
		Sig. (2-tailed)	.	.128	.000	.015
		N	15	15	15	15
	PreTestScore	Correlation Coefficient	.412	1.000	.410	.753**
		Sig. (2-tailed)	.128	.	.129	.001
		N	15	15	15	15
	KMAPostTest	Correlation Coefficient	.845**	.410	1.000	.779**
		Sig. (2-tailed)	.000	.129	.	.001
		N	15	15	15	15
	PostTestScore	Correlation Coefficient	.611*	.753**	.779**	1.000
		Sig. (2-tailed)	.015	.001	.001	.

		N	15	15	15	15
**. Correlation is significant at the 0.01 level (2-tailed).						
*. Correlation is significant at the 0.05 level (2-tailed).						

Table 6.16: Positive Correlation Test with Spearman for Pre-test vs. Post-test scores of the Control group.

Correlation						
			KMAPreTest	**PreTestScore**	**KMAPostTest**	**PostTestScore**
Spearman's rho	KMAPreTest	Correlation Coefficient	1.000	.348	.792**	.416
		Sig. (2-tailed)	.	.158	.000	.086
		N	18	18	18	18
	PreTestScore	Correlation Coefficient	.348	1.000	.364	.981**
		Sig. (2-tailed)	.158	.	.137	.000
		N	18	18	18	18
	KMAPostTest	Correlation Coefficient	.792**	.364	1.000	.438
		Sig. (2-tailed)	.000	.137	.	.069
		N	18	18	18	18
	PostTestScore	Correlation Coefficient	.416	.981**	.438	1.000
		Sig. (2-tailed)	.086	.000	.069	.
		N	18	18	18	18
**. Correlation is significant at the 0.01 level (2-tailed).						

6.1.7 Difference between Pre-test and Post-Test

Wilcoxon tests were employed to investigate the repeated measures of difference between the group. An examination of the findings in Table 6.17 shows that there is a significant difference between the pre-test and post-test academic achievement scores of the students in the experimental group (Z=-2.292, p=.000<.001). The sum of their negative ranks for the experimental group students' academic achievement scores is 7.50, while their sum of positive ranks is 58.50. Given the sum of ranks for the difference scores, the observed difference is in favor of positive ranks, or, in other words, the is a significant improvement of post-test scores as compared to the pre-test scores in the experimental group . Hence, it could be said that the use of the metacognitive activities in learning computer programming significantly increased the academic achievement levels of the experimental group students.

Hypothesis:

H0: µ PostTest - PreTest = µAfter	or	H0: µ PostTest - PreTest = 0	or	H0 :µ D = 0
Ha: PostTest - PreTest > µ After		Ha: µ PostTest - PreTest > 0		Ha :µ D > 0

Significant Level at 0.05 reject the null hypothesis if p-value ≤ 0.05. Since p-value = 0.011 ≤ 0.05, we reject the null hypothesis. At the level 0.05 of significance, on the basis of the results obtained, there is enough evidence to conclude that students in the experimental group significantly improved after the computer programming training together with the MSSNP interaction.

Table 6.17: Wilcoxon Test – Changes in the Pre-test and Post-test for the experimental group

Ranks			
	N	Mean	Sum of

			Rank	Ranks
KMAPostTest - KMAPreTestExp	Negative Ranks	1[a]	7.50	7.50
	Positive Ranks	10[b]	5.85	58.50
	Ties	4[c]		
	Total	15		

a. KMAPostTest < KMAPreTestExp
b. KMAPostTest > KMAPreTestExp
c. KMAPostTest = KMAPreTestExp

Test Statistics[a]

	KMAPostTest - KMAPreTestExp
Z	-2.292[b]
Asymp. Sig. (2-tailed)	.022

a. Wilcoxon Signed Ranks Test
b. Based on negative ranks.

A Wilcoxon signed-rank test showed that four weeks training for both groups elicited a statistically significant change in the knowledge monitoring of the individuals (Z = -2.292b, p = .022) that was significant at the 5% level. An examination of the findings in Table 6.17 shows that there is a significant difference between the pre-test and post-test scores of the experimental group with Z=-2.292, p=.022<.005 and that the p value is significant at the 5% level. The sum of the negative ranks for the experimental group is 7.50, while the sum of the positive ranks is 58.50

Table 6.18: Wilcoxon Test – Changes in the Pre-test and Post-test for the control group

Ranks

		N	Mean Rank	Sum of Ranks
KMAPostTest - KMAPreTest	Negative Ranks	2[a]	2.50	5.00
	Positive Ranks	1[b]	1.00	1.00
	Ties	15[c]		
	Total	18		

a. KMAPostTest < KMAPreTest
b. KMAPostTest > KMAPreTest

c. KMAPostTest = KMAPreTest	
Test Statistics[a]	
	KMAPostTest - KMAPreTest
Z	-1.089[b]
Asymp. Sig. (2-tailed)	.276
a. Wilcoxon Signed Ranks Test	
b. Based on positive ranks.	

An examination result of Table 6.18 shows that there is no difference between the pre-test and post-test scores of the experimental group with Z=-1.089, p=.0276>.005 at the 5% level. The sum of the negative ranks of the control group is 2.5 and the sum of the positive ranks is 1. Given the sum of ranks for the difference in scores, the observed difference is in favor of the positive ranks, or, in other words, the post-test scores of the experimental group with the significant difference between the pre-test and post-test. Hence, it can be concluded from this study that the use of the online instruction in metacognitive strategies in learning the Introductory Computer Programming throughout the four weeks training show the increase of the knowledge monitoring performance of novice programmers.

6.2 Usability Test

Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users. In the context definition of ISO 9241, usability testing is defined as "the extent to which a product can be used by specified users to achieve goals with effectiveness, efficiency and satisfaction in the specified context of use" (Veenendaal, 1998). The Software Usability Measurement Inventory (SUMI) is chosen as the method to measure user acceptance of the prototype system. The five attributes that make up the SUMI that are described by (Kirakowski & Corbett, 1993) as follows:

- Affect – The user's general emotional reaction to the software

- Efficiency – The degree to which users feel that the software assists them in their work
- Helpfulness – The degree to which the software is self-explanatory, as well as more specific things like the adequacy of help facilities and documentation
- Control – Measures the extent to which the user feels in control of the software, as opposed to being controlled by the software, when carrying out the task
- Learnability – Measures the speed and facility with which the user feels that they have been able to master the system, or to learn how to use new features when necessary

6.2.1 Respondents

Ten respondents volunteered to take part in this study. Most of the respondents share the same background, that is, first to third year semester students from the computer science field of study. They were randomly selected from the survey participants. (Nielsen, 2000) indicated that the appropriate size of user per session in usability test is 5 unless there are highly disparate conditions and that it should be conducted with distinct groups of users. Because metacognition is hard to measure and because each person has a different level of metacognitive and cognitive knowledge and skill, having 10 users in a usability study per session would help us to gain an insight for future direction in designing the software solution.

6.2.2 Usability testing procedure

The usability testing of the MSSNP was done in the computer-lab room with each participant being provided with a computer. Ten users who were invited via email agreed to take part in the study. The email stated the details of the testing session information (i.e. time, venue and the objective). The experiment was conducted and administered in the computer-lab room. A verbal instruction was given to the respondents at the beginning of the session to present the SUMI method and to help them understand the objective of the evaluation test. They were also required to become well acquainted with the MSSNP environment. The respondents were not given any instruction concerning how to operate the MSSNP. An average of five minutes was given to the participants to try out the system. They were assisted by the experimenters in respect of any difficulties they encountered with the questionnaire. At the end of the test session, the participants were asked to complete the questionnaire. The questionnaire form was divided into three sections consisting of general information, evaluation criteria (SUMI questions) and suggestions or comments (see Appendix M). The general information section provided the information about the background of the respondents in terms of gender, age and level of study (Diploma or Bachelor Degree), the second section focused on the evaluation criteria of the MSSNP based on the SUMI attributes (affect, efficiency, helpfulness, learnability) and the last section consisted of one open-ended question on suggestions or comments. The sessions lasted about 20 minutes.

6.2.3 Results

The overall scores for the various SUMI attributes are presented in Table 1. In general, the results indicate that user satisfaction with the system is encouraging in that the rating score was better than average and within the desired range of 40 to 60. The "global" in the SUMI scale is used as the benchmark for determining the overall judgement of usability. As shown in Table 6.19, the global score is 60, which exceeds the benchmark score.

Table 6.19: SUMI scores for the MSSNP

Attr.	Global	Efficiency	Affect	Control	Helpfulness	Learnability
Mean	59	62	60	59	57	59

Table 6.20 presents the individual score rated by 10 users for various SUMI subscales. Based on the usability ratings, as shown in Table 6.20, the lowest score was for the "Helpfulness" attribute, which is 56. This might be due to the lack of help facilities and documentation for the software. Overall, the results indicate the desired range of 40 to 60, which shows that users are satisfied with the MSSNP as a supporting tool for learning introductory computer programming. In order to improve the score rating, the design has to emphasize all the attributes of SUMI, especially the "Helpfulness" and "Learnability" attributes.

Table 6.20: Scores of SUMI subscales per user

User	Global	Efficiency	Affect	Control	Helpfulness	Learnability
User 1	63	67	68	61	57	60
User 2	55	54	58	58	52	51
User 3	59	59	52	52	60	70
User 4	61	68	61	65	49	61
User 5	60	67	57	57	68	51
User 6	58	51	65	49	67	57
User 7	60	65	60	60	51	65
User 8	64	70	56	67	65	60
User 9	58	71	57	61	40	62
User 10	58	51	65	59	59	57

Besides SUMI, as part of the usability test, an open-ended question of comments or suggestions for further improvements to the MSSNP presents varied feedback from the participants:

- It would be nice if the MSSNP provides a forum or platform to enable students and lecturers to communicate and discuss the programming method, strategies and planning, etc.
- I think the MSSNP is a good system to make learners aware and apply their metacognitive skills appropriately in learning computer programming.
- From my point of view, the MSSNP can be improved in terms of its look and feel, such as the layout, color as well as the behavior of the dynamic elements, such as menu and buttons.
- It would be better if the MSSNP can be integrated with the compiler or syntax error checking library so that students could post their programming code to check syntax error and the system could provide recommendations or advice concerning how to fix the problems.
- A search engine is something that the MSSNP should consider as a help support feature.

Out of ten participants, four participants gave suggestions for further improvement of the MSSNP in terms of the functionality and the design aspects. One participant felt that the MSSNP would help novices to be aware of their metacognitive skills and apply it appropriately in learning computer programming.

6.3 Summary

The experimental results show that the MSSNP made an overall positive difference in the student interaction approach with the problem-solving environment. We believe that the presence of metacognitive skill and awareness was one of the elements that contributed to the quality of students' attitude towards the problem-solving activity in the MSSNP. The experimental group performed better than the control group in terms of giving more correct answers in the post-test support for this claim. They (experimental group) also presented better time management in problem-solving tasks as well as gave up on fewer problems, which indicates the high level of motivation possessed by them in handling difficult problems. The conducted experiment was quite intricate to execute as it involved system development together with the different materials, activities as well as information analyzed from different sources. Concerning the design, we acknowledge that our experiment has some limitations. The small number of participants in the experiment is one of the strong limitations. Hence, to draw a more definite conclusion concerning the influence and the benefits of the proposed metacognitive activities, another experimental study with a greater number of participants and better control is necessary to overcome the limitations mentioned above.

CHAPTER 7: CONCLUSION

This chapter presents a summary of the research design and interpretation of the important research findings in relation to the research objective. The key findings from the previous chapter are discussed. Several suggestions are provided for possible extensions of this study in the future. Lastly, conclusions are drawn to wrap up the study.

7.1 Summary of the study

This section presents an overview of the study. The summary restates the problems, provides a brief description of how the study was conducted, and presents the major findings in relation to the research objective. The research aims to identify factors in supporting metacognitive activities in learning programming. The literature study revealed that teaching and learning computer programming was difficult for both the learner and the instructor and was indicated as being a universal problem. In Chapter three, we discussed the findings of observational study on the pedagogical approach in teaching and learning computer programming at university. The findings of the survey showed that three types of learning approach are commonly used by the instructor to teach computer programming; cooperative learning, discussion and dialog, and repetition where there are positive attributes and negatives attributes exhibited among the students from each of the pedagogical approaches. We also conducted the survey using the MAI instrument with the following aims:

1. To identify the metacognitive effect towards the learning success of computer programming
2. To identify the novice programmer learning behavior

3. To identify the motivating factor of utilizing the support system in learning computer programming

To investigate this issue in depth, interview sessions were conducted with expert lecturers with the objective of exploring the metacognitive implementation and metacognition awareness in teaching computer programming courses at universities. To gain further understanding on learning behavior of the novice programmer, interview sessions were also conducted with students.

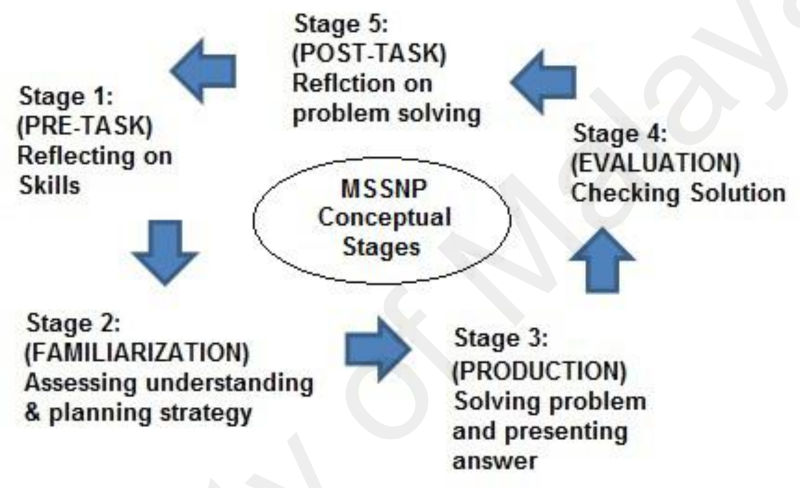


Figure 7.1: The MSSNP Conceptual Stages

As a result of the findings and the literature review, we identify five main stages to represent the metacognitive activities in the MSSNP in supporting novice programmers in learning computer programming, as shown in Figure 7.1. Each has its own objectives as follows:

1. Pre-Task – The main objective of this stage is for student to be stimulated by their own reflection in monitoring their knowledge. It focuses on the past experience in solving problems as well as the performance.
2. Familiarization – The main objective of this phase is for them to reflect on the strategies that may help them in problem-solving activities.

3. Production – The main objective of this stage is to reflect the student's understanding concerning the concept as well as their confidence to solve the problem correctly.
4. Evaluation – This stage only involves the activity of checking the solution provided by the lecturer, which is used as a comparison in studying the student's solution.
5. Post-Task – The opportunity will be given to the students to review their most recent experiences as well as exploring things that happened during the activity of solving the problem.

To support all these activities were support with Semantic Web as the underlying technology of the developed prototype system. With this technology information that relates to both of the skills; cognitive skills and metacognitive skills were expressed in precise form that can be interpreted by machine and ready for software agents to process, share that would enable the application to interoperate on the semantic as well as syntactic level. The developed primary ontology and domain ontology are used as the enabler technology to support the cognitive and metacognitive activities of the proposed system. The research findings are further discussed in the next section.

7.2 Discussion of the findings

This study employed a quantitative approach that combines observation and interview techniques for data collection. Observation and Interview are used to understand complex social processes and capture the essential aspects of a phenomenon from the perspective of the study's participants. Table 7.1 presents the results of this research work based on the mapping of research questions, research objectives and research methods.

Table 7.1: Mapping of research questions, research objectives, methods and results

#	**Research Question (s)**	**Research Objective (s)**	**Research Method (s)**	**Results**
1	How does metacognition affect the learning success of computer programming at university?	To ascertain the effect of metacognition in learning computer programming	○ Questionnaire Survey with Undergraduate Students ○ Interviews with Expert Lecturers ○ Interviews with Students	○ Metacognitive awareness has a significant effect on the successful learning of Introductory Computer Programming at universities ○ Expert Lecturers recognize the importance of metacognition in teaching and learning Computer Programming
2	What is the role of metacognition in teaching and learning of computer programming?			
3	What are the characteristics of the metacognitive instruction and activities to be incorporated in the support system in learning programming?	Identification of the support features for the metacognitive learning environment	○ Questionnaire Survey with Undergraduate Students ○ Interviews with Expert Lecturers ○ Interviews with Students	○ Self-assessment, ○ Knowledge monitoring ○ Self-questioning ○ Graphic organizer ○ Scaffolding ○ Self-directed learning
4	How can a metacognitive support environment benefit novice programmers in learning computer programming?	Development of support features as identified in (2) using Semantic Web technology	○ MSSNP Ontology Design ○ Development of modules of MSSNP	○ An ontology design of MSSNP. ○ A prototype system called Metacognitive Support Environment for Novice Programmers
		Evaluation on the effectiveness of the system prototype and its usability	○ Evaluation on the Usability and Experimental Study of the system before and after treatment	○ Good Usability Evaluation ○ Satisfactory on the effectiveness of the prototype system

These findings are subject to research limitations. Recommendations regarding the findings are provided:

1. **How does metacognition affect the learning success of computer programming at university?**

   This section presents the findings related to the first objective of this research work, Identification of the effect of metacognition in learning computer programming'. The findings from the literature review, survey and in-depth interviews revealed that metacognition is one of the factors that determine the success of the academic achievement of novices, especially in learning computer programming. Consistent with the studies done by (Bernard & Bachu, 2015; Cetin et al., 2014; Mercado & Romo, 2011) and a few other researchers, the findings support the claim that the higher the degree of metacognitive awareness possessed by novice programmers the greater the learning success in Computer Programming subject at university and vice versa. The findings show that students with low metacognitive awareness lack the eight components of metacognitive skills described by (Tobias S. et al., 1999), that is – Declarative knowledge, Procedural knowledge, Planning, Information Management Strategies, Comprehension Monitoring, Debugging Strategies and Evaluation. Declarative knowledge refers to the factual information and knowledge that novices know. It refers to facts or information stored in the memory that are considered static in nature. Declarative Knowledge also referred to as conceptual, propositional or descriptive knowledge, describes things, events, or processes, their attributes, and their relation to each other. Whereby procedural knowledge refers to how novices perform some task, also known as an imperative knowledge. This knowledge is considered as knowledge related to methods, procedures or operations of specific task, whereas procedural is also known as implicit knowledge or know-know.

2. **What is the role of metacognition in teaching and learning of computer programming?**

   From the conducted interviews, the expert lecturers recognized that metacognition is one of the factors that play an important role in the learning success of computer programming learning at university that must happen before, during and after instruction.

3. **What are the characteristics or features of metacognitive instruction to be incorporated in support system in learning programming?**

   This research question concerns the identification of the support features for a metacognitive learning environment to be incorporated in the proposed system as follows:

   - **Scaffolding** – is the support given during the learning process that is tailored to the interactive learning environment that must be embedded in the system
   - **Self-questioning** – is the element that encourages the process of asking and answering while learning
   - **Self-directed learning** – individuals select, manage, and assess their own learning activities
   - **Self-assessment** – the process of having the learners critically reflect upon, and record the progress on their own learning
   - **Graphic Organizer** – is used to organize information during learning

- **Timing** – learning reflection should happen before, during and after instruction

4. **How can a metacognitive support environment benefit novice programmers in learning computer programming?**

   Because we intend to measure the metacognitive state of novice programmers, an experimental study was conducted to determine how effective the proposed support system is in improving novice programmers' metacognitve awareness. The conducted pre-test (before treatment) and post-test (after treatment) show that there are significant changes in terms of the metacognitive behavior of learners measured by the KMA. The conducted usability test on the prototype system also shows an encouraging result.

7.3 Limitations and future research directions

The metacognitive activities proposed in this study are limited in that they only pertain to information about the state of knowledge monitoring of the learner. Other aspects of metacognition have to be overlooked, such as the evaluation of learning, planning strategies and ability, etc. Another flaw is the fact that it does not take the motivational state of the student into consideration, as this is another important factor for reflective and effective learning. There is also a lack of the interactivity in the system in terms of the graphical reification used to represent information (plain graphical). It would be more interesting if the system could use flash animation for a more user interactive experience, and gather feedback from the students regarding the 'look and feel of the system'. Although the study shows that there is a positive difference in terms of the way students interact with the environment regarding problem-solving, with a small number of participants it may not really represent the real situation. Besides forums, collaborative support with peers is something that should be considered for future improvement.

7.4 Contribution of this Research

This research benefits novice programmers in the sense that it provides a metacognitive environment in learning computer programming. The major contributions of this study can be accessed from four perspectives. This research work provides:

- Identification of Novice Programmer’s metacognition awareness towards the learning success in learning Introductory Computer Programming at University
- Identification of the role of metacognition in the learning success of teaching and learning Computer Programming at University
- Development of instructional design for the instruction of metacognition in problem-solving environments that include the activities that relate to the reflection to be performed before as well as after problem-solving attempts
- Evaluation of a fully working system. The system implements the knowledge measurement assessment introduced by Tobias and Everson

The main research question was: Does a metacognitive support environment help novices to improve their metacognition? With the implementation of metacognition activities in the MSSNP, it is expected that the interaction would encourage novice programmers to be more effective and have the potential to:

- encourage students to be more aware in allocating the cognitive resources efficiently in problem-solving activities in learning computer programming
- encourage students to think of appropriate metacognitive strategies in solving problems
- motivate students to use their metacognitive skills in solving problems

7.5 Future work

A wide range of studies was explored during the course of this research; however, these studies were excluded from this dissertation because of the limitations in the length and scope of the thesis. Nevertheless, the benefits of such a literature survey became very apparent, particularly towards the end of this study. There are several points in terms of direction that could be explored due to the limitation of the MSSNP and the insights that were gained from the experiment conducted for further investigation. Thus, we recommend that the following points be explored in future research:

- Improvement of metacognitive model and activities by taking into account the novice programmers' mental model and some additional information that is relevant to metacognition could also be considered, such as strategies selected, the time spent for each activity, novice programmer motivation and provide some mechanism to transform this data into the metacognition information of an individual.
- One of our intentions in this research work is to measure the metacognitive states of an individual, thus, including collaborative activities, such as comparison of knowledge and information of individuals could result in deeper reflection and give a greater observable effect of this study
- Selective reflective activities that rely on the student's metacognitive state would be beneficial and make it even more attractive because a more precise picture to represent the state of student level would mean a more comprehensive set of conditions that are possible to explore. For instance, the activity 'Metacognitive Selection' and the KMA of the student is low, therefore, it is beneficial to offer when the problem is difficult instead of always, as is currently the case.

7.6 Conclusion

This research has demonstrated the significance of metacognition as one of the factors to determine the learning success in computer science education. This research work has also shown the benefits of support tools that balance metacognitive and cognitive tasks. Semantic Web is an enabling technology that facilitates the process of distinguishing the ambiguity on the definition of cognition and metacognition in the development of Metacognitive Support Environment for Novice Programmer in learning Computer Programming. There are other aspects and factors of learning success in problem-solving computer programming and the metacognitive development of novice programmers that are still unexplored and that could possibly be taken into account. The embedding of the social component through collaborative activities would be a promising direction for making such an environment more effective and meaningful to students. As such this research work should be of interest to a broad range of researchers including those interested in educational psychology and computer science education, and we have an expectation that it can give motivation for people to take it further. To conclude, this research work supports the utilization of metacognition in teaching and learning in the Computer Science field for effective and learning success.

References

Aleven, V., Roll, I., & Koedinger, K. R. (2012). Progress in Assessment and Tutoring of Lifelong Learning Skills. *Adaptive technologies for training and education*, 69.

Aleven, V. A., & Koedinger, K. R. (2002). An effective metacognitive strategy: Learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive Science*, *26*(2), 147-179.

Allert, J. (2004). *Learning style and factors contributing to success in an introductory computer science course.* Paper presented at the Advanced Learning Technologies, 2004. Proceedings. IEEE International Conference on.

Amulya, J. (2004). What is reflective practice. *Center for Reflective Community Practice, Massachusetts Institute of Technology, Cambridge, MA.[Online]. Available at: http://www. itslifejimbutnotasweknowit. org. uk/files/whatisreflectivepractic e. pdf [Accessed 15 April 2009]*.

Angelo, T. A., & Cross, K. P. (1993). Classroom assessment techniques.

Apiola, M., Tedre, M., & Oroma, J. O. (2011). *Improving programming education in Tanzania: Teachers' and students' perceptions.* Paper presented at the Frontiers in Education Conference (FIE), 2011.

Askell-Williams, H., Lawson, M. J., & Skrzypiec, G. (2012). Scaffolding cognitive and metacognitive strategy instruction in regular class lessons. *Instructional Science*, *40*(2), 413-443.

Azevedo, R., & Aleven, V. A. (2013). *International handbook of metacognition and learning technologies* (Vol. 26): Springer.

Azevedo, R., Cromley, J. G., Winters, F. I., Moos, D. C., & Greene, J. A. (2005). Adaptive human scaffolding facilitates adolescents' self-regulated learning with hypermedia. *Instructional science*, *33*(5-6), 381-412.

Baas, D., Castelijns, J., Vermeulen, M., Martens, R., & Segers, M. (2014). The relation between Assessment for Learning and elementary students' cognitive and metacognitive strategy use. *British Journal of Educational Psychology*.

Baker, L., & Brown, A. L. (1984). Metacognitive skills and reading. *Handbook of reading research*, *1*(353), V394.

Banks, A. P., & Millward, L. J. (2007). Differentiating Knowledge in Teams: The Effect of Shared Declarative and Procedural Knowledge on Team Performance. *Group Dynamics: Theory, Research, and Practice, 11*(2), 95.

Bannert, M., & Reimann, P. (2012). Supporting self-regulated hypermedia learning through prompts. *Instructional Science, 40*(1), 193-211.

Bardach, E. (2011). *Practical guide for policy analysis: the eightfold path to more effective problem solving*: CQ press.

Berges, M. (2015). *Investigating Novice Programming Abilities with the Help of Psychometric Assessment.* Paper presented at the Society for Information Technology & Teacher Education International Conference.

Bergin, S., Reilly, R., & Traynor, D. (2005). *Examining the role of self-regulated learning on introductory programming performance.* Paper presented at the Proceedings of the first international workshop on Computing education research.

Berliner, D. C., & Calfee, R. C. (1996). *Handbook of educational psychology*: Macmillan Library Reference USA, Simon & Schuster Macmillan.

Bernard, M., & Bachu, E. (2015). Enhancing the Metacognitive Skill of Novice Programmers Through Collaborative Learning *Metacognition: Fundaments, Applications, and Trends* (pp. 277-298): Springer.

Berners-Lee, T., Fischetti, M., & Foreword By-Dertouzos, M. L. (2000). *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*: HarperInformation.

Bickhard, M. H. (2013). Scaffolding and self scaffolding: Central aspects of development. *LT Winegar, Sc]. Valsiner (Eds), Chz'ldreri'sd U, 610*, 33-52.

Bielaczyc, K., Pirolli, P. L., & Brown, A. L. (1995). Training in self-explanation and self-regulation strategies: Investigating the effects of knowledge acquisition activities on problem solving. *Cognition and instruction, 13*(2), 221-252.

Borkowski, J. G., Carr, M., & Pressley, M. (1987). "Spontaneous" strategy use: Perspectives from metacognitive theory. *Intelligence, 11*(1), 61-75.

Boud, D., Keogh, R., & Walker, D. (2013). Promoting reflection in learning A modeli. *Boundaries of adult learning, 1,* 32.

Boyle, J. R., Rosen, S. M., & Forchelli, G. (2014). Exploring metacognitive strategy use during note-taking for students with learning disabilities. *Education 3-13*(ahead-of-print), 1-20.

Brand-Gruwel, S., & Stadtler, M. (2011). Solving information-based problems: Evaluating sources and information. *Learning and Instruction, 21*(2), 175-179.

Bransford, J. (2000). *How people learn: Brain, mind, experience, and school*: National Academies Press.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (2000). How people learn: Washington, DC: National Academy Press.

Briñol, P., & DeMarree, K. G. (2012). *Social metacognition*: Psychology Press.

Brophy, J. E. (2013). *Motivating students to learn*: Routledge.

Brown, A. (1987). Metacognition, executive control, self-regulation and other more mysterious mechanisms. *Metacognition, motivation, and understanding*, 65-116.

Brown, A. (1987). Metacognition, executive control, self-regulation, and other more mysterious mechanisms. *Metacognition, motivation, and understanding*, 65-116.

Brown, A. L. (1980). Metacognitive development and reading. *Theoretical issues in reading comprehension*, 453-481.

Brown, J. S., Collins, A., & Newman, S. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. *Knowing, learning, and instruction: Essays in honor of Robert Glaser*, 487.

Bulu, S. T., & Pedersen, S. (2012). Supporting problem-solving performance in a hypermedia learning environment: The role of students' prior knowledge and metacognitive skills. *Computers in Human Behavior, 28*(4), 1162-1169.

Camahalan, F. M. G. (2006). Effects of a Metacognitive Reading Program on the Reading Achievement and Metacognitive Strategies of Students with Cases of Dyslexia. *Reading improvement, 43*(2), 77-93.

Carr, M., Kurtz, B. E., Schneider, W., Turner, L. A., & Borkowski, J. G. (1989). Strategy acquisition and transfer among American and German children: Environmental influences on metacognitive development. *Developmental Psychology, 25*(5), 765.

Carver, S., & Mayer, R. (1988). Learning and transfer of debugging skills: Applying task analysis to curriculum design and assessment. *Teaching and Learning Computer Programming: Multiple Research Perspectives. RE Mayer. Hillsdale, NJ, Lawrence Erlbaum Associates*, 259-297.

Cella, M., Swan, S., Medin, E., Reeder, C., & Wykes, T. (2014). Metacognitive awareness of cognitive problems in schizophrenia: exploring the role of symptoms and self-esteem. *Psychological medicine, 44*(03), 469-476.

Cetin, I., Sendurur, E., & Sendurur, P. (2014). Assessing the Impact of Meta-Cognitive Training on Students' Understanding of Introductory Programming Concepts. *Journal of Educational Computing Research, 50*(4), 507-524.

Chambliss, M. J., & Calfee, R. C. (1998). *Textbooks for learning: Nurturing children's minds*: Blackwell Publishers Malden, Massachusetts.

Choi, M. J., & Jeong, D. Y. (2013). A Study on the Effect of Metacognition to the Information-Seeking Behavior of Undergraduate Students. *Journal of the Korean Society for Library and Information Science, 47*(2), 75-101.

Cleary, T. J., Durning, S. J., Gruppen, L. D., Hemmer, P. A., & Artino Jr, A. R. (2013). Self-regulated learning. *Oxford textbook of medical education*, 465.

Costa, C. J., Aparicio, M., & Cordeiro, C. (2012). *A solution to support student learning of programming.* Paper presented at the Proceedings of the Workshop on Open Source and Design of Communication.

Coull, N. J. (2008). *SNOOPIE: development of a learning support tool for novice programmers within a conceptual framework.* University of St Andrews.

Coull, N. J., & Duncan, I. M. (2011). Emergent requirements for supporting introductory programming. *Innovation in Teaching and Learning in Information and Computer Sciences, 10*(1), 78-85.

Cross, D. R., & Paris, S. G. (1988). Developmental and instructional analyses of children's metacognition and reading comprehension. *Journal of Educational Psychology, 80*(2), 131.

Dabarera, C., Renandya, W. A., & Zhang, L. J. (2014). The impact of metacognitive scaffolding and monitoring on reading comprehension. *System, 42*, 462-473.

Davidson, J. E., Deuser, R., & Sternberg, R. J. (1994). The role of metacognition in problem solving.

Davis, E. A. (2000). Scaffolding students' knowledge integration: Prompts for reflection in KIE. *International Journal of Science Education, 22*(8), 819-837.

Davis, E. A. (2014). Scaffolding Learning.

De Backer, L., Van Keer, H., & Valcke, M. (2012). Fostering university students' metacognitive regulation through peer tutoring. *Procedia-Social and Behavioral Sciences, 69*, 1594-1600.

Dechant, E. (2013). *Understanding and teaching reading: An interactive model*: Routledge.

Deejring, K. (2015). The Validation of Web-based Learning Using Collaborative Learning Techniques and a Scaffolding System to Enhance Learners' Competency in Higher Education. *Procedia-Social and Behavioral Sciences, 174*, 34-42.

Deitel, H. M., Deitel, P. J., & Nieto, T. (1994). *C++ how to program* (Vol. 4): Prentice Hall Englewood cliffs.

Deutskens, E., De Ruyter, K., Wetzels, M., & Oosterveld, P. (2004). Response rate and response quality of internet-based surveys: An experimental study. *Marketing letters, 15*(1), 21-36.

Donker, A., De Boer, H., Kostons, D., van Ewijk, C. D., & Van der Werf, M. (2014). Effectiveness of learning strategy instruction on academic performance: A meta-analysis. *Educational Research Review, 11*, 1-26.

Dorça, F. A., Lima, L. V., Fernandes, M. A., & Lopes, C. R. (2013a). Comparing strategies for modeling students learning styles through reinforcement learning in adaptive and intelligent educational systems: An experimental analysis. *Expert Systems with Applications, 40*(6), 2092-2101.

Dorça, F. A., Lima, L. V., Fernandes, M. A., & Lopes, C. R. (2013b). A new approach to discover students learning styles in adaptive educational systems. *Revista Brasileira de Informática na Educação, 21*(01), 76.

Drucker, P. (2000). Need to know: Integrating e-learning with high velocity value chains. *A Delphi Group White Paper*, 1-12.

Elshout-Mohr, M., Meijer, J., van Daalen-Kapteijns, M., & Meeus, W. (2003). *A self-report inventory for metacognition related to academic tasks.* Paper presented at the 10th Conference of the European Association for Research on Learning and Instruction (EARLI) Padova, Italy.

Ennis, R. H. (1985). A Logical Basis for Measuring Critical Thinking Skills. *Educational leadership, 43*(2), 44-48.

Ernst*, J., & Monroe, M. (2004). The effects of environment-based education on students' critical thinking skills and disposition toward critical thinking. *Environmental Education Research, 10*(4), 507-522.

Ertmer, P. A., & Newby, T. J. (1996). The expert learner: Strategic, self-regulated, and reflective. *Instructional science, 24*(1), 1-24.

Eysenck, M. W., Ellis, A. W., Hunt, E. B., & Johnson-Laird, P. N. E. (1994). *The Blackwell dictionary of cognitive psychology*: Basil Blackwell.

Falkner, K., & Palmer, E. (2009). *Developing authentic problem solving skills in introductory computing classes.* Paper presented at the ACM SIGCSE Bulletin.

Feyzi-Behnagh, R., Azevedo, R., Legowski, E., Reitmeyer, K., Tseytlin, E., & Crowley, R. S. (2014). Metacognitive scaffolds improve self-judgments of accuracy in a medical intelligent tutoring system. *Instructional science, 42*(2), 159-181.

Flavell, J. (1987). Speculations about the nature and development of metacognition. En FE Weinert & RH Kluwe (Eds.), Metacognition, Motivation and Understanding (pp. 21-29). Hillside: New Jersey: Lawrence Erlbaum Associates.[Links].

Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American psychologist, 34*(10), 906.

Foley, J. D., Van Dam, A., Feiner, S. K., Hughes, J. F., & Phillips, R. L. (1994). *Introduction to computer graphics* (Vol. 55): Addison-Wesley Reading.

Ford, C. L., & Yore, L. D. (2012). Toward convergence of critical thinking, metacognition, and reflection: Illustrations from natural and social sciences, teacher education, and classroom practice *Metacognition in science education* (pp. 251-271): Springer.

Francom, G. M. (2010). Teach me how to learn: principles for fostering students' self-directed learning skills. *International Journal of Self-Directed Learning, 7*(1), 29-44.

Frensch, P. A., & Funke, J. (2014). *Complex problem solving: The European perspective*: Psychology Press.

Frith, C. D. (2012). The role of metacognition in human social interactions. *Philosophical Transactions of the Royal Society B: Biological Sciences, 367*(1599), 2213-2223.

Gaddis, S. E. (1998). How to design online surveys. *Training & Development, 52*(6), 67-71.

Gan, M. J., & Hattie, J. (2014). Prompting secondary students' use of criteria, feedback specificity and feedback levels during an investigative task. *Instructional Science, 42*(6), 861-878.

Garner, R. (1990). When children and adults do not use learning strategies: Toward a theory of settings. *Review of educational research, 60*(4), 517-529.

Garner, R., & Alexander, P. A. (1989). Metacognition: Answered and unanswered questions. *Educational Psychologist, 24*(2), 143-158.

Garner, S. (2007). A program design tool to help novices learn programming. *ICT: Providing choices for learners and learning. Proceedings ascilite Singapore*.

Garrison, D. R. (2011). *E-learning in the 21st century: A framework for research and practice*: Taylor & Francis.

Garrison, D. R., & Akyol, Z. (2013). Toward the development of a metacognition construct for communities of inquiry. *The Internet and Higher Education, 17*, 84-89.

Ghaleb, F., Daoud, S., Hasna, A., ALJa'am, J. M., El-Seoud, S. A., & El-Sofany, H. (2006). E-learning model based on semantic web technology. *International Journal of Computing & Information Sciences, 4*(2), 63-71.

Giora, R. (1996). Language comprehension as structure building. *Journal of pragmatics, 26*(3), 417-436.

Goh, C. C., & Hu, G. (2014). Exploring the relationship between metacognitive awareness and listening performance with questionnaire data. *Language Awareness, 23*(3), 255-274.

Graesser, A. C., Singer, M., & Trabasso, T. (1994). Constructing inferences during narrative text comprehension. *Psychological review, 101*(3), 371.

Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P., & Kreuz, R. (1999). AutoTutor: A simulation of a human tutor. *Cognitive Systems Research, 1*(1), 35-51.

Greene, J. A., Robertson, J., & Costa, L. C. (2011). Assessing self-regulated learning using think-aloud methods. *Handbook of self-regulation of learning and performance*, 313-328.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition, 5*(2), 199-220.

Grüninger, M., & Fox, M. S. (1995). Methodology for the Design and Evaluation of Ontologies.

Guarino, N. (1998). *Formal ontology in information systems: proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy* (Vol. 46): Ios Pr Inc.

Gumbel, E. J. (2012). *Statistics of extremes*: Courier Corporation.

Halpern, D. F. (2002). *Thought and knowledge: An introduction to critical thinking*: Routledge.

Halpern, D. F. (2014). *Critical Thinking Across the Curriculum: A Brief Edition of Thought & Knowledge*: Routledge.

Hannafin, M., Land, S., & Oliver, K. (1999). Open learning environments: Foundations, methods, and models. *Instructional-design theories and models: A new paradigm of instructional theory, 2*, 115-140.

Harb, J. N., Durrant, S. O., & Terry, R. E. (1993). Use of the Kolb learning cycle and the 4MAT system in engineering education. *Journal of Engineering Education, 82*(2), 70-77.

Hartman, H. J. (2001). Developing students' metacognitive knowledge and skills *Metacognition in learning and instruction* (pp. 33-68): Springer.

Havenga, M. (2011). *Problem-solving processes in computer programming: a case study*. Paper presented at the Southern African Computer Lecturers' Association (SACLA) Conference Proceedings.

Havenga, M., Mentz, E., Breed, B., Govender, D., Govender, I., Dignum, F., & Dignum, V. (2012). *A case study regarding teachers' problem-solving activities and approaches towards computer programming in diverse learning* *environments*. Paper presented at the 3rd International Conference on Society and Information Technologies ICSIT, Orlando (USA), 25-28 March, 2012.

Haynie, J. M., Shepherd, D. A., & Patzelt, H. (2012). Cognitive adaptability and an entrepreneurial task: The role of metacognitive ability and feedback. *Entrepreneurship Theory and Practice, 36*(2), 237-265.

Hazzan, O., Lapidot, T., & Ragonis, N. (2011). *Guide to Teaching Computer Science: An Activity-Based Approach*: Springer.

Henderson, P. (1987). Modern introductory computer science. *ACM SIGCSE Bulletin, 19*(1), 183-190.

Hill, M., & Greive, C. (2011). The Potential to Promote Social Cohesion, Self-Efficacy and Metacognitive Activity: A Case Study of Cross-Age Peer-Tutoring. *TEACH Journal of Christian Education, 5*(2), 10.

Hoc, J. M. (1990). *Psychology of programming*: Academic Pr.

Horrocks, I., & Sattler, U. (2001). *Ontology reasoning in the SHOQ (D) description logic*. Paper presented at the IJCAI.

Horton, S. V., Lovitt, T. C., & Bergerud, D. (1990). The effectiveness of graphic organizers for three classifications of secondary students in content area classes. *Journal of Learning Disabilities, 23*(1), 12-22.

Hu, M., Winikoff, M., & Cranefield, S. (2012). *Teaching novice programming using goals and plans in a visual notation*. Paper presented at the Proceedings of the Fourteenth Australasian Computing Education Conference-Volume 123.

Hughes, A. J. (2015). Impact of Online Self-regulated Professional Development on Technology and Engineering Educators Metacognitive Awareness.

Ifenthaler, D. (2012). Determining the effectiveness of prompts for self-regulated learning in problem-solving scenarios. *Journal of Educational Technology & Society, 15*(1), 38–52.

Ismail, M. A., Yaacob, M., Kareem, S. A., & Halim, A. H. A. Domain Specific Ontology Construction: Falbo’s Approach.

Ismail, M. N., Azilah, N., Naufal, U., & Kelantan, U. T. M. C. (2010). Instructional strategy in the teaching of computer programming: a need assessment analyses. *TOJET, 9*(2), 125-131.

Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010). The effects of mind mapping with cooperative learning on programming performance, problem solving skill and metacognitive knowledge among computer science students. *Journal of Educational Computing Research, 42*(1), 35-61.

Jacobse, A. E., & Harskamp, E. G. (2012). Towards efficient measurement of metacognition in mathematical problem solving. *Metacognition and Learning, 7*(2), 133-149.

Jones, B. F., & Idol, L. (2013). *Dimensions of thinking and cognitive instruction*: Routledge.

Joseph, L. M., Alber-Morgan, S., Cullen, J., & Rouse, C. (2015). The effects of self-questioning on reading comprehension: A literature review. *Reading & Writing Quarterly*(ahead-of-print), 1-22.

Kalyanpur, A. (2004). SWOOP (Semantic Web Ontology Overview and Perusal): MindSwap, Maryland Information and Network Dynamics Lab Semantic Web Agents Project, http://www. mindswap. org/people/, v2.

Katz, Y. J. (2002). Attitudes affecting college students' preferences for distance learning. *Journal of Computer Assisted Learning, 18*(1), 2-9.

Kessler, C. M., & Anderson, J. R. (1986). Learning flow of control: Recursive and iterative procedures. *Human-Computer Interaction, 2*(2), 135-166.

Kim, M. C., & Hannafin, M. J. (2011). Scaffolding problem solving in technology-enhanced learning environments (TELEs): Bridging research and theory with practice. *Computers & Education, 56*(2), 403-417.

Kirakowski, J., & Corbett, M. (1988). *Measuring user satisfaction.* Paper presented at the Proceedings of the Fourth Conference of the British Computer Society on People and computers IV.

Kirakowski, J., & Corbett, M. (1993). SUMI: The software usability measurement inventory. *British journal of educational technology, 24*(3), 210-212.

Kölling, M., & Rosenberg, J. (1996). *Blue—a language for teaching object-oriented programming.*

Koss, M. (2002). Apollo-The user guide 1.0. *Knowledge Media Institute, The Open University.*

Kramarski, B., & Mevarech, Z. R. (2003). Enhancing mathematical reasoning in the classroom: The effects of cooperative learning and metacognitive training. *American Educational Research Journal*, *40*(1), 281-310.

Kranch, D. A. (2012). Teaching the novice programmer: A study of instructional sequences and perception. *Education and Information Technologies, 17*(3), 291-313.

Kuhn, D., & Dean, J., David. (2004). Metacognition: A bridge between cognitive psychology and educational practice. *Theory into practice, 43*(4), 268-273.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). *A study of the difficulties of novice programmers.* Paper presented at the ACM SIGCSE Bulletin.

Lai, G. (2008). Examining the effects of selected computer-based scaffolds on preservice teachers' levels of reflection as evidenced in their online journal writing.

Latva-karjanmaa, R. (2001). Mediated Learning in Virtual Learning Environments.

Laurillard, D. (2013). *Rethinking university teaching: A conversational framework for the effective use of learning technologies*: Routledge.

Lee, D. M. C., Rodrigo, M. M. T., d Baker, R. S., Sugay, J. O., & Coronel, A. (2011). Exploring the relationship between novice programmer confusion and achievement *Affective Computing and Intelligent Interaction* (pp. 175-184): Springer.

Lee, H. W., Lim, K. Y., & Grabowski, B. L. (2010). Improving self-regulation, learning strategy use, and achievement with metacognitive feedback. *Educational Technology Research and Development, 58*(6), 629-648.

Lee, M. J., & Ko, A. J. (2011). *Personifying programming tool feedback improves novice programmers' learning.* Paper presented at the Proceedings of the seventh international workshop on Computing education research.

Lee, T. B., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American, 284*(5), 34-43.

Liffick, B. W., & Aiken, R. (1996). *A novice programmer's support environment.* Paper presented at the ACM SIGCSE Bulletin.

Liffick, B. W., & Aiken, R. (1996). A novice programmer's support environment. *ACM SIGCSE Bulletin, 28*(SI), 49-51.

Lin, X. (2001). Designing metacognitive activities. *Educational Technology Research and Development, 49*(2), 23-40.

Linn, M. C., & Clancy, M. J. (1992). The case for case studies of programming problems. *Communications of the ACM, 35*(3), 121-132.

Livingston, J. (1996). Effects of metacognitive instruction on strategy use of college students. *Unpublished manuscript, State University of New York at Buffalo*.

Madureira, A., Gomes, S., Cunha, B., Pereira, J., Santos, J., & Pereira, I. (2014). *Prototype of an Adaptive Decision Support System for Interactive Scheduling with MetaCognition and User Modeling Experience.* Paper presented at the Nature and Biologically Inspired Computing (NaBIC), 2014 Sixth World Congress on.

Mangione, G. R., Gaeta, M., Orciuoli, F., & Salerno, S. (2010). *A Semantic Metacognitive Learning Environment*.

Markova, G., & Legerstee, M. (2013). Implicit Confusions in Metacognition. *Infant and Child Development, 22*(1), 105-107.

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

Mathan, S. A., & Koedinger, K. R. (2005). Fostering the intelligent novice: Learning from errors with metacognitive tutoring. *Educational Psychologist, 40*(4), 257-265.

Mayer, R. E. (2003). *Learning and instruction*: Prentice Hall.

Mayer, R. E. (2011). *Applying the science of learning*: Pearson/Allyn & Bacon Boston.

Mayer, R. E. (2013). *Teaching and learning computer programming: Multiple research perspectives*: Routledge.

Mayer, R. E., Dyck, J. L., & Vilberg, W. (1986). Learning to program and learning to think: what's the connection? *Communications of the ACM, 29*(7), 605-610.

Mazzoni, G., & Nelson, T. O. (2014). *Metacognition and cognitive neuropsychology: Monitoring and control processes*: Psychology Press.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B. D., . . . Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin, 33*(4), 125-180.

McGill, T. J., & Volet, S. E. (1997). A conceptual framework for analysing students' knowledge of programming. *Journal of Research on Computing in Education, 29*(3), 276-297.

McGilly, K. (1996). *Classroom lessons: Integrating cognitive theory and classroom practice*: The MIT Press.

McIlraith, S. A., Son, T. C., & Zeng, H. (2001). Semantic web services. *Intelligent Systems, IEEE, 16*(2), 46-53.

McNie, E. C. (2007). Reconciling the supply of scientific information with user demands: an analysis of the problem and review of the literature. *Environmental Science & Policy, 10*(1), 17-38.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education, 23*(3), 239-264.

Mercado, C. A. A., & Romo, L. I. S. (2011). Mental Models, Strategies and Metacognition in Programming, and it's relation to Verbal Protocols as a learning mechanism3. *Analysis of State-of-the-Art Solutions for Personalised Learning Support*, 32.

Merriam, S. B., Caffarella, R. S., & Baumgartner, L. M. (2012). *Learning in adulthood: A comprehensive guide*: John Wiley & Sons.

Metallidou, P., & Efklides, A. (2002). The effects of general success-related beliefs and specific metacognitive experiences on causal attributions *Trends and prospects in motivation research* (pp. 325-347): Springer.

Metcalfe, J. E., & Shimamura, A. P. (1994). *Metacognition: Knowing about knowing*: The MIT Press.

Miller, T. M., & Geraci, L. (2011). Training metacognition in the classroom: The influence of incentives and feedback on exam predictions. *Metacognition and learning, 6*(3), 303-314.

Mishra, S., & Director, C. (2013). Learning and Technology: Emerging Trends to Democratise Education.

Molenaar, I., Roda, C., van Boxtel, C., & Sleegers, P. (2012). Dynamic scaffolding of socially regulated learning in a computer-based learning environment. *Computers & education, 59*(2), 515-523.

Molenaar, I., van Boxtel, C. A., & Sleegers, P. J. (2011). Metacognitive scaffolding in an innovative learning arrangement. *Instructional Science, 39*(6), 785-803.

Moreno, A., Myller, N., Sutinen, E., & Ben-Ari, M. (2004). *Visualizing programs with Jeliot 3*. Paper presented at the Proceedings of the working conference on Advanced visual interfaces.

Murray, T., Stephens, L., Woolf, B. P., Wing, L., Xu, X., & Shrikant, N. (2013). Supporting social deliberative skills online: the effects of reflective scaffolding tools *Online Communities and Social Computing* (pp. 313-322): Springer.

Nett, U. E., Goetz, T., Hall, N. C., & Frenzel, A. C. (2012). Metacognitive Strategies and Test Performance: an experience sampling analysis of students' learning behavior. *Education Research International, 2012*.

Newman, R. S. (2002). How self-regulated learners cope with academic difficulty: The role of adaptive help seeking. *Theory into Practice, 41*(2), 132-138.

Nickerson, R. S. (2012). *Aspects of rationality: Reflections on what it means to be rational and whether we are*: Psychology Press.

Nielsen, J. (2000). Why you only need to test with 5 users: Alertbox.

Noy, N., & McGuinness, D. L. (2001). Ontology Development 101. *Knowledge Systems Laboratory, Stanford University*.

Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R. W., & Musen, M. A. (2001). Creating semantic web contents with protege-2000. *IEEE intelligent systems, 16*(2), 60-71.

Oechsle, R., & Schmitt, T. (2002). Javavis: Automatic program visualization with object and sequence diagrams using the java debug interface (jdi) *Software Visualization* (pp. 176-190): Springer.

Ornstein, P. A., Baker-Ward, L., & Naus, M. (1988). The development of mnemonic skill. *Memory development: Universal changes and individual differences*, 31-50.

Özsoy, G. (2011). An investigation of the relationship between metacognition and mathematics achievement. *Asia Pacific Education Review, 12*(2), 227-235.

Palinscar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and instruction, 1*(2), 117-175.

Papert, S., & Solomon, C. (1971). Twenty things to do with a computer.

Paris, S. G., & Winograd, P. (1990). How metacognition can promote academic learning and instruction. *Dimensions of thinking and cognitive instruction, 1*, 15-51.

Paul, R. (1992). Critical thinking: What, why, and how. *New Directions for Community Colleges*, *1992*(77), 3-24.

Pishghadam, R., & Khajavy, G. H. (2013). Intelligence and metacognition as predictors of foreign language achievement: A structural equation modeling approach. *Learning and Individual Differences*, *24*, 176-181.

Poli, R., & Koza, J. (2014). *Genetic Programming*: Springer.

Polikoff, I. (2003). Ontology Tool Support (Ontology Development Lifecycle and Tools): TopQuadrant Technology Briefing, TopQuadrant, Inc, Alexandria, VA, USA.

Puntambekar, S., & Du Boulay, B. (1997). Design and Development of MIST: A System To Help Students Develop Metacognition. *Journal of Educational Computing Research, 16*(1), 1-35.

Ramaha, N. T., & Ismail, W. M. F. W. (2012). Assessment of Learner's Motivation In Web Based E-Learning. *International Journal of Scientific & Engineering Research, 3*(8).

Rea, L. M., & Parker, R. A. (2012). *Designing and conducting survey research: A comprehensive guide*: John Wiley & Sons.

Reder, L. M. (2014). *Implicit memory and metacognition*: Psychology Press.

Reusser, K. (1993). Tutoring systems and pedagogical theory: Representational tools for understanding, planning, and reflection in problem solving. *Computers as cognitive tools, 1*, 143-177.

Reynolds, M. (2011). Reflective practice: origins and interpretations. *Action Learning: Research and Practice, 8*(1), 5-13.

Rist, R. S. (1996). Teaching Eiffel as a first language. *Journal of Object-Oriented Programming, 9*(1), 30-41.

Roberts, M. J., & Erdos, G. (1993). Strategy selection and metacognition. *Educational Psychology, 13*(3-4), 259-266.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137-172.

Roll, I., Aleven, V., McLaren, B. M., & Koedinger, K. R. (2011). Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction, 21*(2), 267-280.

Rosch, E. (1999). Principles of categorization. *Concepts: core readings*, 189-206.

Rosenzweig, C., Krawec, J., & Montague, M. (2011). Metacognitive Strategy Use of Eighth-Grade Students With and Without Learning Disabilities During Mathematical Problem Solving A Think-Aloud Analysis. *Journal of learning disabilities, 44*(6), 508-520.

Rouse, C. A., Alber-Morgan, S. R., Cullen, J. M., & Sawyer, M. (2014). Using Prompt Fading to Teach Self-Questioning to Fifth Graders with LD: Effects on Reading Comprehension. *Learning Disabilities Research & Practice, 29*(3), 117-125.

Rum, M., Nurulain, S., & Ismail, M. A. (2014). *Ontology development of metacognitive support system for novice programmers (MSSNP).* Paper presented at the Technology Management and Emerging Technologies (ISTMET), 2014 International Symposium on.

Rum, S. N. M., & Ismail, M. A. (2014). Usability Evaluation of Metacognitive Support System for Novice Programmers (MSSNP) using SUMI. *Asian Journal of Education and e-Learning, 2*(5).

Sampson, D. G., Lytras, M. D., Wagner, G., & Diaz, P. (2004). Ontologies and the Semantic Web for E-learning. *Educational Technology & Society, 7*(4), 26-28.

Sandi-Urena, S., Cooper, M., & Stevens, R. (2012). Effect of cooperative problem-based lab instruction on metacognition and problem-solving skills. *Journal of Chemical Education, 89*(6), 700-706.

Santos, J. R. A. (1999). Cronbach’s alpha: A tool for assessing the reliability of scales. *Journal of extension, 37*(2), 1-5.

Scardamalia, M., & Bereiter, C. (1985). Fostering the development of self-regulation in children's knowledge processing. *Thinking and learning skills: Research and open questions, 2*, 563-577.

Scheid, K. (1993). *Helping Students Become Strategic Learners. Guidelines for Teaching. Cognitive Strategy Training Series*: ERIC.

Scheier, M. F., Carver, C. S., Clark, M., & Fiske, S. (2014). *Cognition, affect, and self-regulation.* Paper presented at the Affect and Cognition: 17th Annual Carnegie Mellon Symposium on Cognition.

Schellenberg, S., Negishi, M., & Eggen, P. (2011). The Effects of Metacognition and Concrete Encoding Strategies on Depth of Understanding in Educational Psychology. *Teaching Educational Psychology, 7*(2), 17-24.

Schellings, G. L., van Hout-Wolters, B. H., Veenman, M. V., & Meijer, J. (2013). Assessing metacognitive activities: the in-depth comparison of a task-specific questionnaire with think-aloud protocols. *European journal of psychology of education, 28*(3), 963-990.

Scherer, R., & Tiemann, R. (2012). Factors of problem-solving competency in a virtual chemistry environment: The role of metacognitive knowledge about strategies. *Computers & Education, 59*(4), 1199-1214.

Schneider, W. (1985). Developmental trends in the metamemory-memory behavior relationship: An integrative review. *Metacognition, cognition, and human performance, 1*, 57-109.

Schonlau, M., Ronald Jr, D., & Elliott, M. N. (2002). *Conducting research surveys via e-mail and the web*: Rand Corporation.

Schraw, G. (1998). Promoting general metacognitive awareness. *Instructional science, 26*(1-2), 113-125.

Schraw, G. (2006). Knowledge: Structures and processes. *Handbook of educational psychology, 2*, 245-260.

Schraw, G., & Dennison, R. S. (1994). Assessing metacognitive awareness. *Contemporary educational psychology, 19*(4), 460-475.

Schraw, G., & Gutierrez, A. P. (2015). Metacognitive Strategy Instruction that Highlights the Role of Monitoring and Control Processes *Metacognition: Fundaments, Applications, and Trends* (pp. 3-16): Springer.

Schraw, G., & Moshman, D. (1995). Metacognitive theories. *Educational psychology review, 7*(4), 351-371.

Schraw, G., & Sperling Dennison, R. (1994). Assessing metacognitive awareness. *Contemporary educational psychology, 19*, 460-460.

Scott, K. M. (2014). Taking over someone else's e-learning design: challenges trigger change in e-learning beliefs and practices. *Research in Learning Technology, 22*.

Scott, M. J., & Ghinea, G. (2013). Educating programmers: A reflection on barriers to deliberate practice. *Learning and Teaching in STEM Disciplines*.

Semerari, A., Cucchi, M., Dimaggio, G., Cavadini, D., Carcione, A., Battelli, V., . . . Ronchi, P. (2012). The development of the Metacognition Assessment Interview: Instrument description, factor structure and reliability in a non-clinical sample. *Psychiatry research, 200*(2), 890-895.

Slavin, R. E., & Davis, N. (2006). Educational psychology: Theory and practice.

Soloway, E. (1984). *A cognitively-based methodology for designing languages/environments/methodologies.* Paper presented at the ACM SIGPLAN Notices.

Soloway, E., & Spohrer, J. C. (1989). *Studying the novice programmer*: Lawrence Erlbaum Hillsdale, NJ.

Soloway, E., & Spohrer, J. C. (2013). *Studying the novice programmer*: Psychology Press.

Son, L. K., & Simon, D. A. (2012). Distributed learning: data, metacognition, and educational implications. *Educational Psychology Review, 24*(3), 379-399.

Song, J. (2013). Knowing the knowing: Exploring the relationship between metacognition and intelligence.

Sternberg, R. J., & Frensch, P. A. (2014). *Complex problem solving: Principles and mechanisms*: Psychology Press.

Stewart, P. W., & Hadley, K. (2014). INVESTIGATING THE RELATIONSHIP BETWEEN VISUAL IMAGERY, METACOGNITION AND MATHEMATICS PEDAGOGICAL CONTENT KNOWLEDGE. *Journal of the International Society for Teacher Education, 18*(1).

Stojanovic, L., Staab, S., & Studer, R. (2001). *eLearning based on the Semantic Web*.

Stojanovic, L., Stojanovic, N., & Volz, R. (2002). *Migrating data-intensive web sites into the semantic web.* Paper presented at the Proceedings of the 2002 ACM symposium on Applied computing.

Stull, A. T., & Mayer, R. E. (2007). Learning by doing versus learning by viewing: Three experimental comparisons of learner-generated versus author-provided graphic organizers. *Journal of educational psychology, 99*(4), 808.

Su, A., Yang, S. J., Hwang, W. Y., Huang, C. S., & Tern, M. Y. (2014). Investigating the role of computer-supported annotation in problem-solving-based teaching: An empirical study of a Scratch programming pedagogy. *British Journal of Educational Technology, 45*(4), 647-665.

Sure, Y., Angele, J., & Staab, S. (2003). OntoEdit: Multifaceted inferencing for ontology engineering *Journal on Data Semantics I* (pp. 128-152): Springer.

Thiede, K. W., Griffin, T. D., Wiley, J., & Redford, J. S. (2009). Metacognitive monitoring during and after reading. *Handbook of metacognition in education*, 85-106.

Thompson, V. A., Prowse Turner, J. A., & Pennycook, G. (2011). Intuition, reason, and metacognition. *Cognitive psychology, 63*(3), 107-140.

Thota, N., & Whitfield, R. (2009). *Use of CALMS to enrich learning in introductory programming courses.* Paper presented at the Proceedings of the 17th International Conference on Computers in Education.

Tobias, S., & Everson, H. (1995). Development and Validation of an Objective Measure of Metacognition.

Tobias, S., Everson, H., Laitusis, V., & Fields, M. (1999). *Metacognitive knowledge monitoring: Domain specific or general.* Paper presented at the annual meeting of the Society for the Scientific Study of Reading, Montreal.

Tobias, S., Everson, H. T., & Laitusis, V. (1999). Towards a Performance Based Measure of Metacognitive Knowledge Monitoring: Relationships with Self-Reports and Behavior Ratings.

Tsai, M. J. (2009). The model of strategic e-learning: Understanding and evaluating student e-learning from metacognitive perspectives. *Educational Technology & Society, 12*(1), 34-48.

Tunmer, W. E., & Bowey, J. A. (1984). Metalinguistic awareness and reading acquisition. *Metalinguistic awareness in children: Theory, research, and implications*, 144-168.

Utting, I., Tew, A. E., McCracken, M., Thomas, L., Bouvier, D., Frye, R., . . . Sorva, J. (2013). *A fresh look at novice programmers' performance and their teachers' expectations.* Paper presented at the Proceedings of the ITiCSE working group reports conference on Innovation and technology in computer science education-working group reports.

Uysal, M. P. (2014). Improving First Computer Programming Experiences: The Case of Adapting a Web-Supported and Well-Structured Problem-Solving Method to a Traditional Course. *Contemporary Educational Technology, 5*(3).

Van den Boom, G., Paas, F., van Merriënboer, J. J. G., & Van Gog, T. (2004). Reflection prompts and tutor feedback in a web-based learning environment: Effects on students' self-regulated learning competence. *Computers in Human Behavior, 20*(4), 551-567.

Van der Meij, J., & de Jong, T. (2011). The effects of directive self-explanation prompts to support active processing of multiple representations in a simulation-based learning environment. *Journal of Computer Assisted Learning, 27*(5), 411-423.

Van Heijst, G., Schreiber, A. T., & Wielinga, B. J. (1997). Using explicit ontologies in KBS development. *International Journal of Human Computer Studies, 46*, 183-292.

Van Zile-Tamsen, C. (1994). The role of motivation in metacognitive self-regulation. *Unpublished manuscript, State University of New York at Buffalo*.

Van Zile-Tamsen, C. (1996). *Metacognitive self-regulation and the daily academic activities of college students.* State University of New York at Buffalo.

Veenman, M. V. (2011). Alternative assessment of strategy use with self-report instruments: a discussion. *Metacognition and Learning, 6*(2), 205-211.

Veenman, M. V., Van Hout-Wolters, B. H., & Afflerbach, P. (2006). Metacognition and learning: Conceptual and methodological considerations. *Metacognition and learning, 1*(1), 3-14.

Visser, P. S., Krosnick, J. A., Marquette, J., & Curtin, M. (1996). Mail surveys for election forecasting? An evaluation of the Columbus Dispatch poll. *Public Opinion Quarterly, 60*(2), 181-227.

Vohs, J., Lysaker, P., Francis, M., Hamm, J., Buck, K., Olesek, K., . . . Liffick, E. (2014). Metacognition, social cognition, and symptoms in patients with first episode and prolonged psychoses. *Schizophrenia research, 153*(1), 54-59.

Volet, S. E. (1991). Modelling and coaching of relevant metacognitive strategies for enhancing university students' learning. *Learning and Instruction, 1*(4), 319-336.

Vygotskiĭ, L. L. S. (1978). *Mind in society: The development of higher psychological processes*: Harvard university press.

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001). *Ontology-based integration of information-a survey of existing approaches.* Paper presented at the IJCAI-01 workshop: ontologies and information sharing.

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001). *Ontology-based integration of information-a survey of existing approaches*.

Weinstein, C. E. (1988). *Assessment and training of student learning strategies*: Springer.

Weinstein, C. E., & Mayer, R. E. (1983). *The Teaching of Learning Strategies.* Paper presented at the Innovation abstracts.

Willer, D., Eisenberg, M., & Sadzewicz, J. *Using Metacognition to Improve Information Literacy Skills.* Paper presented at the The Second European Conference on Information Literacy (ECIL).

Williams, P. E., & Hellman, C. M. (2004). Differences in self-regulation for online learning between first-and second-generation college students. *Research in Higher Education, 45*(1), 71-82.

Wilson, N. S., & Bai, H. (2010). The relationships and impact of teachers' metacognitive knowledge and pedagogical understandings of metacognition. *Metacognition and Learning, 5*(3), 269-288.

Winne, P. H. (2011). A cognitive and metacognitive analysis of self-regulated learning. *Handbook of self-regulation of learning and performance*, 15-32.

Wu, L., & Looi, C.-K. (2011). Design of Agent Prompts as Scaffolding for Productive Reflection in an Intelligent Learning Environment. *International Journal of Information Technology, 17*(2).

Wu, L., Valcke, M., & Van Keer, H. (2012). Validation of a Chinese version of metacognitive awareness of reading strategies inventory. *Egitim Arastirmalari-Eurasian Journal of Educational Research, 12*(48), 117-134.

Xie, K., & Bradshaw, A. C. (2008). Using question prompts to support ill-structured problem solving in online peer collaborations. *International Journal of Technology in Teaching and Learning, 4*(2), 148-165.

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.

Zile-Tamsen, V., & Marie, C. (1996). Metacognitive self-regulation and the daily academic activities of college students.

Zimmerman, B. J. (1990). Self-regulated learning and academic achievement: An overview. *Educational psychologist, 25*(1), 3-17.

Zimmerman, B. J., & Kitsantas, A. (2014). Comparing students' self-discipline and self-regulation measures and their prediction of academic achievement. *Contemporary Educational Psychology, 39*(2), 145-155.

Zimmerman, B. J., & Schunk, D. H. (2001). Reflections on theories of self-regulated learning and academic achievement. *Self-regulated learning and academic achievement: Theoretical perspectives, 2*, 289-307.

List of Publications and Papers Presented

Journal

- Usability Evaluation of Metacognitive Support System for Novice Programmer using SUMI (Asian Journal of Education and E-learning (AJEEL)
- Metacognitive Awareness Assessment and Introductory Computer Programming Course Achievement at University, The International Arab Journal of Information Technology (Indexed by ISI)
- Metacognitive Support features in Computer Assisted Learning for Novice Programmers, Journal of Computer Assisted Learning, Siti Nurulain Mohd Rum, Maizatul Akmar Ismail (Indexed by ISI)
- A Semantic Web Approach to the Design of a Metacognitive Support Tool for Novice Programmers, Journal of Semantic Web and Information System (Indexed by ISI)

Proceeding

- Siti Nurulain Mohd Rum and Ismail M.A (2014), Ontology Development of Metacognitive Support System for Novice Programmers (MSSNP), International Symposium on Technology Management and Emerging Technologies (ISTMET), 27th - 29th May 2014, Bandung, Indonesia, IEEE (*SCOPUS-Cited Publication*)
- Mohd Rum, S.N and Ismail, M.A (2013), Promoting Metacognitive Learning Environment in Supporting Novice Programmer Using Semantic Web, International Conference on Emerging Trends in Computer Science & Information Technology, Kuala Lumpur, pp45-48 (*Non-ISI/Non-SCOPUS Cited Publication*)
- A Framework Design of Metacognitive Support System for Novice Programmers (MSSNP) PROCEEDINGS OF KNOWLEDGE MANAGEMENT INTERNATIONAL CONFERENCE (KMICE) 2014 (*ISI-Cited Publication)*
- Mohd Rum, S.N and Ismail, M.A (2014), Development of a Metacognitive Support System for Novice Programmers (MSSNP) Using the Semantic Web (ScienceAsia)

Appendix A

Consent Request Letter

Dear Sir,

I am a Phd candidate from the University of Malaya, and am writing to request permission to conduct a study during your class session in Introduction to Computer Programming. The objective of this study is to get further understanding on how a pedagogical approach assists students during the Introductory Computer Programming Class.

In order to complete this study, observation on several different class sessions is required. Any information obtained in connection with this study will remain confidential and will only be disclosed with your permission. No names or information on the individual will be recorded as I am only interested in group results. Such a report will be used for research purposes only.

Your consideration in this matter will be greatly appreciated,

Thank You.

Yours sincerely,
Siti Nurulain Mohd Rum (Mrs.)
PhD Candidate,
Faculty of Computer Science and Information Technology,
University of Malaya.
snurulain@siswa.um.edu.my

Appendix B
MAI Questionnaire

Introduction

This questionnaire is part of the research project in Information Science, University of Malaya.
You have been chosen to take part in this research and I will be very grateful if you can spend some of your precious time completing the given questionnaire. All the gathered information is confidential and will only be used for academic purposes.

“Metacognition is generally defined as the activity of monitoring and controlling one's cognition. In learning programming, this skill is essential to successful learning because its enables individual to better manage their programming skills and to determine weaknesses that can be corrected by constructing a new skills. The objective of this study is to relate the metacognitive awareness with the fundamental programming course achievement in Universities. Kindly look over the questionnaire and spare some of your time to complete the survey. I wish to thank you for your participation in this study”

Yours Faithfully;
Siti Nurulain Mohd Rum
Email: snurulain@um.edu.my
Department of Information Science
University Of Malaya

Section 1 – Personal and Education Background

1.	Your Age?	○ 20 – 25 years ○ 26 – 30 years ○ 31 – 35 years
2.	Gender	○ Male ○ Female
3.	University	▼
4.	Year/Semester of Study	
5	GPA For Introductory Programming Course Regardless the language used	2.0 – 2.49 ○ 2.5 – 2.99 ○ 3.0 – 3.49 ○ 3.5 – 4.00 ○

Section 2 – Metacognitive Awareness Assessment This section consists of 52 statements that relate to your experience in learning the Introductory Computer Programming course at your university, and you are required to rate each as True or False

No	**Declarative Knowledge**	**Statement**	
DCL1	I understand my intellectual strengths and weaknesses.	○ Yes	○ No
DCL2	I know what kind of information is most important to learn.	○ Yes	○ No
DCL3	I am good at organizing information.	○ Yes	○ No
DCL4	I know what the teacher expects me to learn.	○ Yes	○ No
DCL5	I am good at remembering information.	○ Yes	○ No
DCL6	I have control over how well I learn.	○ Yes	○ No
DCL7	I am a good judge of how well I understand something.	○ Yes	○ No

DCL8	I learn more when I am interested in the topic.	○ Yes	○ No
No	**Procedural Knowledge**	**Statement**	
P1	I try to use strategies that have worked in the past.	○ Yes	○ No
P2	I have a specific purpose for each strategy I use.	○ Yes	○ No
P3	I am aware of what strategies I use when I study.	○ Yes	○ No
P4	I find myself using helpful learning strategies automatically.	○ Yes	○ No
No	**Conditional Knowledge**	**Statement**	
CDL1	I learn best when I know something about the topic.	○ Yes	○ No
CDL2	I use different learning strategies depending on the situation.	○ Yes	○ No
CDL3	I can motivate myself to learn when I need to.	○ Yes	○ No
CDL4	I use my intellectual strengths to compensate for my weaknesses.	○ Yes	○ No
CDL5	I know when each strategy I use will be most effective.	○ Yes	○ No
No	**Planning**	**Statement**	
PL1	I pace myself while learning in order to have enough time.	○ Yes	○ No
PL2	I think about what I really need to learn before I begin a task.	○ Yes	○ No
PL3	I set specific goals before I begin a task.	○ Yes	○ No
PL4	I ask myself questions about the material before I begin.	○ Yes	○ No
PL5	I think of several ways to solve a problem and choose the best one.	○ Yes	○ No
PL6	I read instructions carefully before I begin a task.	○ Yes	○ No
PL7	I organize my time to best accomplish my goals.	○ Yes	○ No
No	**Information Management Strategies**	**Statement**	
IMS1	I slow down when I encounter important information.	○ Yes	○ No
IMS2	I consciously focus my attention on important	○ Yes	○ No

	information.		
IMS3	I focus on the meaning and significance of new information.	○ Yes	○ No
IMS4	I create my own examples to make information more meaningful.	○ Yes	○ No
IMS5	I draw pictures or diagrams to help me understand while learning.	○ Yes	○ No
IMS6	I try to translate new information into my own words.	○ Yes	○ No
IMS7	I use the organizational structure of the text to help me learn	○ Yes	○ No
IMS8	I ask myself if what I'm reading is related to what I already know.	○ Yes	○ No
IMS9	I try to break studying down into smaller steps.	○ Yes	○ No
IMS10	I focus on overall meaning rather than specifics.	○ Yes	○ No
No	**Comprehension Monitoring**	**Statement**	
CM1	I ask myself periodically if I am meeting my goals.	○ Yes	○ No
CM2	I consider several alternatives to a problem before I answer.	○ Yes	○ No
CM3	I ask myself if I have considered all options when solving a problem.	○ Yes	○ No
CM4	I periodically review to help me understand important relationships.	○ Yes	○ No
CM5	I find myself analyzing the usefulness of strategies while I study.	○ Yes	○ No
CM6	I find myself pausing regularly to check my comprehension.	○ Yes	○ No
CM7	I ask myself questions about how well I am doing while learning something new.	○ Yes	○ No

No	Debugging Strategies	Statement	
DBG1	I ask others for help when I don’t understand something.	○ Yes	○ No
DBG2	I change strategies when I fail to understand.	○ Yes	○ No
DBG3	I re-evaluate my assumptions when I get confused.	○ Yes	○ No
DBG4	I stop and go back over new information that is not clear.	○ Yes	○ No
DBG5	I stop and reread when I get confused.	○ Yes	○ No
No	**Evaluation**	**Statement**	
EVL1	I know how well I did once I finish a test.	○ Yes	○ No
EVL2	I ask myself if there was an easier way to do things after I finish a task.	○ Yes	○ No
EVL3	I summarize what I’ve learned after I finish.	○ Yes	○ No
EVL4	I ask myself how well I accomplished my goals once I’m finished.	○ Yes	○ No
EVL5	I ask myself if I have considered all options after I solve a problem.	○ Yes	○ No
EVL6	I ask myself if I learned as much as I could have once I finish a task.	○ Yes	○ No
	Strategies used for learning Computer Programming	**Answer options**	
i.	Rehearsal	☐	
ii.	Elaboration	☐	
iii.	Organization	☐	
iv.	Critical Thinking	☐	
v.	Metacognition Regulation	☐	
vi.	Time & Study Environment	☐	
vii.	Effort regulation	☐	
viii.	Peer Learning	☐	
ix.	Help-Seeking	☐	
	What is the		
i.	Efficiency	☐	
ii.	Affect	☐	
iii.	Effort	☐	
iv.	Goal Orientation	☐	

v.	Helpfulness	☐
vi.	Learnability	☐
vii.	Control	☐
viii.	Self-regulation	☐

Appendix C

Distribution of Respondents

Distribution by Gender

Gender	Frequency	Percentage (%)
Male	62	37.8
Female	102	62.2
Total	164	100

Distribution by GPA

GPA	Frequency	Percentage (%)
2.00 – 2.49	44	26.83
2.50 – 2.99	56	34.15
3.00 - 3.49	40	24.39
3.50 – 4.00	24	14.63
Total	164	100

Academic year of study

GPA	Frequency	Percentage (%)
Year 2	75	45.73
Year 3	38	23.17
Year 4	35	21.34
Year 5	10	6.1
Year 6	6	3.66
Total	164	100

Appendix D

Results of Correlation Coefficient between all variables

Y	X	r	r^2	t	Pr(>|t|)
GPA	MAI	0.8226	0.6767	18.4136	0.0000
GPA	P	0.4387	0.1925	6.2140	0.0000
GPA	DL	0.7358	0.5413	13.8274	0.0000
GPA	CDL	0.6134	0.3762	9.8843	0.0000
GPA	PL	0.7061	0.4986	12.6917	0.0000
GPA	IMS	0.6882	0.4737	12.0747	0.0000
GPA	CM	0.7025	0.4935	12.5629	0.0000
GPA	DBG	0.6023	0.3627	9.6027	0.0000
GPA	EVL	0.5679	0.3225	8.7816	0.0000
GPA	KC	0.7483	0.5599	14.3562	0.0000
GPA	RC	0.8224	0.6763	18.3987	0.0000
MAI	P	0.6186	0.3827	10.0207	0.0000
MAI	DL	0.8638	0.7462	21.8251	0.0000
MAI	CDL	0.7519	0.5654	14.5175	0.0000
MAI	PL	0.8469	0.7172	20.2709	0.0000
MAI	IMS	0.7848	0.6159	16.1173	0.0000
MAI	CM	0.8639	0.7463	21.8309	0.0000
MAI	DBG	0.6545	0.4283	11.0177	0.0000
MAI	EVL	0.7317	0.5354	13.6633	0.0000
MAI	KC	0.9257	0.8570	31.1568	0.0000
MAI	RC	0.9764	0.9534	57.5703	0.0000
P	DL	0.4464	0.1993	6.3503	0.0000
P	CDL	0.4717	0.2225	6.8085	0.0000
P	PL	0.4368	0.1908	6.1811	0.0000
P	IMS	0.4024	0.1619	5.5945	0.0000
P	CM	0.5232	0.2737	7.8140	0.0000
P	DBG	0.4059	0.1647	5.6521	0.0000
P	EVL	0.3645	0.1329	4.9825	0.0000
P	KC	0.6698	0.4486	11.4797	0.0000
P	RC	0.5523	0.3051	8.4328	0.0000
DL	CDL	0.6192	0.3834	10.0363	0.0000
DL	PL	0.7193	0.5174	13.1786	0.0000
DL	IMS	0.5926	0.3511	9.3628	0.0000
DL	CM	0.7939	0.6303	16.6180	0.0000
DL	DBG	0.4806	0.2310	6.9761	0.0000
DL	EVL	0.5728	0.3281	8.8934	0.0000
DL	KC	0.9071	0.8228	27.4249	0.0000
DL	RC	0.7878	0.6206	16.2789	0.0000
CDL	PL	0.6276	0.3938	10.2591	0.0000
CDL	IMS	0.5709	0.3259	8.8508	0.0000
CDL	CM	0.5649	0.3191	8.7130	0.0000
CDL	DBG	0.4589	0.2106	6.5732	0.0000
CDL	EVL	0.4760	0.2266	6.8886	0.0000
CDL	KC	0.8289	0.6870	18.8584	0.0000
CDL	RC	0.6815	0.4644	11.8528	0.0000

PL	IMS	0.5884	0.3462	9.2618	0.0000
PL	CM	0.7752	0.6010	15.6212	0.0000
PL	DBG	0.5579	0.3112	8.5562	0.0000
PL	EVL	0.5663	0.3207	8.7459	0.0000
PL	KC	0.7347	0.5398	13.7840	0.0000
PL	RC	0.8615	0.7423	21.6000	0.0000
IMS	CM	0.5442	0.2961	8.2553	0.0000
IMS	DBG	0.5533	0.3061	8.4541	0.0000
IMS	EVL	0.5896	0.3477	9.2918	0.0000
IMS	KC	0.6548	0.4288	11.0268	0.0000
IMS	RC	0.8199	0.6723	18.2308	0.0000
CM	DBG	0.5071	0.2572	7.4888	0.0000
CM	EVL	0.5277	0.2785	7.9076	0.0000
CM	KC	0.7810	0.6099	15.9145	0.0000
CM	RC	0.8521	0.7262	20.7263	0.0000
DBG	EVL	0.4684	0.2194	6.7472	0.0000
DBG	KC	0.5446	0.2966	8.2648	0.0000
DBG	RC	0.6866	0.4714	12.0204	0.0000
EVL	KC	0.6087	0.3705	9.7643	0.0000
EVL	RC	0.7451	0.5551	14.2175	0.0000
KC	RC	0.8347	0.6967	19.2897	0.0000

Appendix E
Interview Result with Expert Lecturer

Interview Result with Respondent 1 and Respondent 2

No	Questions description	Respondent 1	Respondent 2
Q1	Have you ever heard of metacognition?	“Haven’t heard”	“Yes, but not sure what it is”
	After reading the definition, would you describe what metacognition is?	“Cognition awareness”	“Monitoring of cognition and it awareness”
Q2	How is metacognition important for a lecturer in teaching?	“It is important and the lecturer must think about their own thinking”	“Lecturer must think hard, it is important for effective learning”
Q3	Do you think metacognition is an important factor in computer science education success? Why?	“It is important for students to be independent learners, therefore nurturing students with metacognition skills is something that is crucial”	“We have to educate students to act and think”
Q4	How are metacognitive skills been taught to your students for learning improvement?	“Encourage students to ask questions”	“Ask students to read books or find information from the Internet and write down their thinking”
Q5	How do you apply metacognition in your own teaching?	“Prepare well before teaching, monitor myself during teaching to ensure message can be conveyed effectively”	“Listening to student and try to answer any questions in simplest way that can be easily understood”
Q6	When you teach, what is important to you?	“Students are able to extract the important information given during class and able write it down”	“Know what students need to know and learn, how the best to teach it and how to tell if they have learned it”
Q7	What are you thinking about when you are teaching?	“Always keep this old proverb, ‘if you feed a person a fish, then the person eats only then. But if you teach a person how to catch fish, then the person will eat well in the future’, students need to be independent learners”	“Asking myself, do I provide enough information for them to understand?”
Q8	Before you teach, what do	“Analyze and confirm	“Do preparation for

	you usually do?	the lesson content as well as appropriate teaching material”	class the day before and try to come early a few minutes in the classroom”
Q9	What do you think about before you teach?	Prepare handouts and material	Prepare learning content
Q10	After you teach, what do you usually do?	“Add content information that was insufficiently provided and clarified in last session and prepare lesson for the following session.”	“Evaluate the strengths and weaknesses of own teaching”
Q11	Common problems faced by novice programmers?	“Lack of skill for analyzing problem and unable to apply similar or past problem solutions into new problem”	“Most student fail to understand the concept of programming in bigger picture”

Interview Results with Respondent 3 and Respondent 4

No	Questions description	Respondent 3	Respondent 4
Q1	Have you ever heard of metacognition?	Never	Never
	After reading the article given, would you describe the definition of metacognition ?	“knowledge of cognition”	“Applying learning theory”
Q2	How is metacognition important for a lecturer in teaching?	“Educator must understand their own teaching processes”	“Lecturer must be aware of their own thoughts, emotions, and behaviors”
Q3	Do you think metacognition is as an important factor in computer science education success? Why?	“Ask students to take notes and prepare a lesson plan”	“Suggest YouTube /Internet for students to be independent learner”
Q4	How have metacognitive skills been taught to your students for learning improvement?	“Give more assignments”	“Let students know the whole process of problem-solving”
Q5	How do you apply metacognition in your own teaching?	“Interaction with Students”	“Writing,” “Discussing,”
Q6	When you teach, what is important to you?	“Everything must be in order in the classroom	“Effective Classroom environment”
Q7	What are you thinking about when you are teaching?	“How student learns best”	
Q8	Before you teach, what do you usually do?	“Write outline for discussion in classroom”	“Make notes and review activities”
Q9	What do you think about before you teach?	“Plan for outline and material for discussion during	“Lesson-plans”

		lesson”	
Q10	After you teach, what do you usually do?	“assess teaching and plan for next round teaching”	“Spend another 15 or 30 minutes after class to answer questions from students”
Q11	Common problems faced by novice programmers?	Lack of skills in analyzing problems	Difficult to understand programming concept

Interview Result with Respondent 5

No	Question description	Respondent 5
Q1	Have you ever heard of metacognition?	“No, haven’t heard”
	After reading the definition, would you describe what metacognition is?	“Knowledge Monitoring”
Q2	How is metacognition important for a lecturer in teaching?	“Lecturers must be aware of their own thoughts, emotions and behavior”
Q3	Do you think metacognition is an important factor in computer science education success? Why?	“Yes, it’s crucial for effective teaching and learning”
Q4	How do you teach metacognitive skills to your students to improve their learning?	“Students need to be taught how to act and think”
Q5	How do you apply metacognition in your own teaching?	“Thinking and questioning with students”
Q6	When you teach, what is important to you?	“Students’ requirements are first priority.”
Q7	What are you thinking about when you are teaching?	“How to make students learn and understand”
Q8	Before you teach, what do you usually do?	“Design and plan lessons”
Q9	What do you think about before you teach?	“How to finish the lesson within the given time”
Q10	After you teach, what do you usually do?	“Assess/evaluate student learning by giving short quizzes”

Appendix F
Interview Result with Novice Programmer

Respondents' answers for Question 1 to Question 4

Respondent	Question 1	Question 2	Question 3	Question 4
R1	No	"To stay motivated, study with others through knowledge sharing session"	Internet, Books	Translate information into my own words
R2	No	"Try to discover the context on why such knowledge is important and exchange knowledge with others"	Internet, Materials given by instructor	Draw pictures or diagrams
R3	No	"Not motivated to learn this subject, very difficult to understand "	Internet	Write summary note
R4	Yes	"Just enjoy coding that can keep me motivated "	Internet, Books	make stories and pictures
R5	No	"Study with peer"	Internet	used organizational structure

Respondents' answer for Question 5 to Question 7

Respondent	Question 5	Question 6	Question 7
R1	Seek help from the social media like forum and Facebook	Yes, usually it will take about 2 - 4 hours a day	Always communicates with lecturer to get to know strengths and weakness
R2	Seek the help of instructor	No	Do more exercises and seek help from instructor
R3	Seek the help of instructor or someone that has good knowledge in programming	No	Check quizzes and test paper and comparing with others
R4	Study with friends and comparing answer from quiz, test and assignment given by lecturer	No	Do tutorial and check answer with instructor

Appendix G

Pre-Test Question (Part1)

Participant ID: ___________________

Pre Test (PART 1)

Instructions- Read each problem given and provide answer in each box. Please do not solve the problems.

1. Write a pseudocode that reads two numbers and multiplies them together and prints out their product

Question: Based on the problem given, do you think you can provide the answer?

[ ] Yes [ ] No

2. Write a pseudocode that performs the following. Ask a user to enter a number. If the number is between 0 and 10, write the word blue. If the number is between 10 and 20, write the word red. If the number is between 20 and 30, write the word green

Question: Based on the problem given, do you think you can provide the answer?

[ ] Yes [ ] No

3. Write a pseudo code to print all multiples of 5 between 1 and 100 (including both 1 and 100)

Question: Based on the problem given, do you think you can provide the answer?

[ ] Yes [ ] No

4. Write a pseudo code that will count all the even numbers up to a user defined stopping point. For example, say we want to see the first 5 even numbers starting from 0.

Question: Based on the problem given, do you think you can provide the answer?

[ ] Yes [ ] No

5. Write pseudocode that performs the following. Ask a user to enter Year of birth and printout user generation as follows:

IF year 1946 – 1964 = 'Baby Boomers'

IF year 1965 – 1984 = ‘Generation X’

IF year 1985 – 2000 = ‘Generation Y’

IF year > 2000 = ‘Generation Z’

Question: Based on the problem given, do you think you can provide the answer?

[ ] Yes [ ] No

Appendix H

Pre-Test Question (Part 2)

Participant ID: ___________________________________

Pre-Test – PART 2

Instruction: You are given 1 hour to solve all the problems. The experimenter will alert you when the time is up. You are alloweed to solve the problems given in any order.

.

Kindly read the box given below before you start.

Pseudocode: is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a “text-based” detail (algorithmic) design tool. The rules of Pseudocode are reasonably straightforward. All Statements showing “dependency” are to be indented. These include WHILE, DO, FOR, IF and SWITCH.

6. Write pseudocode that reads two numbers and multiplies them together and prints out their product

7. Write a pseudocode that performs the following. Ask a user to enter a number. If the number is between 0 and 10, write the word blue. If the number is between 10 and 20, write the word red. If the number is between 20 and 30, write the word green

8. Write a pseudocode to print all multiples of 5 between 1 and 100 (including both 1 and 100)

9. Write a pseudocode that will count all the even numbers up to a user defined stopping point. For example, say we want to see the first 5 even numbers starting from 0.

The first 5 even numbers are 0,2,4,6,8.
The first 8 even numbers are 0,2,4,6,8,10,12,16

```
READ count
```

10. Write a pseudocude that performs the following. Ask a user to enter Year of birth and printout user generation as follows:

```
IF year 1946 – 1964 = ‘Baby Boomers’
IF year 1965 – 1984 = ‘Generation X’
IF year 1985 – 2000 = ‘Generation Y’
IF year > 2000 = ‘Generation Z’
```

```
  DISPLAY “ Please enter year of birth”
  RECEIVE  birth year from KEYBOARD
  IF year >=  1946 and year <= 1964
      DISPLAY ‘Baby Boomers’
  IF year >=  1945 and year <= 1984
      DISPLAY ‘Generation X’
   IF year >=  1985 and year <= 2000
      DISPLAY ‘Generation Y’
IF year >=  2000
      DISPLAY ‘Generation Z’
```

Appendix I

Post-Test Question (Part 1)

Participant ID: ___________________

Pre Test – PART 1

Instructions- Read each problem given and provide answer in each box. Please don't solve the problems

1.Read in a number representing a temperature in degrees Celsius and write it out as a value in degrees Fahrenheit. If the Celsius value is C, then the Fahrenheit value, F, is calculated as follows: F = (9 / 5) * C + 32.

Question: Based on the problem given, do you think you can provide the answer?

[ ] Yes [ ] No

2.Read in 10 numbers and write out the average of those number

Question: Based on the problem given, do you think you can provide the answer?

[ ] Yes [ ] No

3.Store and process the race times of the finalists in a 100 m sprint so that the winner's time is output. Given all the 10 times as follows [10.23, 10.1,10.29,9,9,10.12,10.34,9.99,9.58]

Question: Based on the problem given, do you think you can provide the answer?

[ ] Yes [ ] No

4.Write pseudocode that reads in the three numbers and writes them all in sorted order

Question: Based on the problem given, do you think you can provide the answer?

[ ] Yes [ ] No

5.Write pseudocode that will count all the even numbers up to a user defined stopping point. For example, say we want to see the first 5 even numbers starting from 0. Well, we know that the even numbers are 0, 2, 4, etc. The first 5 even numbers are 0, 2, 4, 6, 8. The first 8 even numbers are 0, 2, 4, 6, 8 ,10 ,12, 16

Question : Based on the problem given, do you think you can provide the answer?

[ ] Yes [ ] No

Appendix J

Post-Test Question (Part 2)

Participant ID: ___________________________________

Post-Test – PART 2

Instruction: You are given 1 hour to solve all the problems. The experimenter will alert you when the time is up. You are allowed to solve the problems given in any order.

Kindly read the box given below before you start.

Pseudocode: is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool. The rules of Pseudocode are reasonably straightforward. All Statements showing "dependency" are to be indented. These include WHILE, DO, FOR, IF and SWITCH.

1. Read in a number representing a temperature in degrees Celsius and write it out as a value in degrees Fahrenheit. If the Celsius value is C, then the Fahrenheit value, F, is calculated as follows: F = (9 / 5) * C + 32.

2. Read in 10 numbers and write out the average of those numbers

3. Store and process the race times of the finalists in a 100 m sprint so that the winner's time is output. Given all the 10 times as follows [10.23, 10.1,10.29,9,9,10.12,10.34,9.99,9.58]

4. Write a pseudocode that reads in the three numbers and writes them all in sorted order

5. Write a pseudocode that will count all the even numbers up to a user defined stopping point.
   For example, say we want to see the first 5 even numbers starting from 0.
   well, we know that the even numbers are 0, 2, 4, etc.
   The first 5 even numbers are 0, 2, 4, 6, 8.
   The first 8 even numbers are 0, 2, 4, 6, 8 ,10 ,12, 16

Appendix K

<u>Sample Schema Answer for Pre-Test (Part 2)</u>

1. Write a pseudocode that reads two numbers and multiplies them together and prints out their product

```
RECEIVE num1, num2 FROM KEYBOARD
SET multi to num1 * num2
SEND multi TO DISPLAY
```

2. Write a pseudocode that performs the following. Ask a user to enter a number. If the number is between 0 and 10, write the word blue. If the number is between 10 and 20, write the word red. If the number is between 20 and 30, write the word green

```
DISPLAY “Please enter a number”
RECEIVE numColor FROM KEYBOARD
IF  numColor > 0 AND numColor <=10
    DISPLAY Blue
ELSEIF numColor > 10 and numColor  <=20
    DISPLAY Red
ELSE numColor > 10 and numColor <=30
    DISPLAY Green
```

3. Write a pseudocode to print all multiples of 5 between 1 and 100 (including both 1 and 100)

```
SET x to 1
WHILE ( x < 20)
 DISPLAY x
 SET x = x * 5
```

4. Write a pseudocode that will count all the even numbers up to a user defined stopping point. For example, say we want to see the first 5 even numbers starting from 0.

   The first 5 even numbers are 0,2,4,6,8.
   The first 8 even numbers are 0,2,4,6,8,10,12,16

5. Write a pseudo code to find the factorial of N where the value of N is input through the keyboard.?

```
fact=1
n=input(“Please, Enter a number\n”)
c=1
while(c<=n):
fact=fact*c
c=c+1
print “The factorial of “, n , “ is “, fact
```

Appendix L

<u>Sample Schema Answer for Post-Test (Part 2)</u>

1.Read in a number representing a temperature in degrees Celsius and write it out as a value in degrees Fahrenheit. If the Celsius value is C, then the Fahrenheit value, F, is calculated as follows: F = (9 / 5) * C + 32.

Using our pseudocode, this would be written as follows:

```
RECEIVE c FROM KEYBOARD
SET f TO ( 9 / 5 ) * c + 32
SEND f TO DISPLAY
```

2.Read in 10 numbers and write out the average of those number

This would be presented as follows:

```
 SET total TO 0
 SET count TO 0

 WHILE count < 10 DO
      RECEIVE nextInput FROM KEYBOARD
       SET total TO 0
       SET count TO count + 1
END WHILE
SEND total / 10 to display
```

3.Store and process the race times of the finalists in a 100 m sprint so that the winner's time is output. Given all the 10 times as follows:
[10.23, 10.1,10.29,9,9,10.12,10.34,9.99,9.58]

```
 SET allTimes TO [10.23, 10.1,10.29,9,9,10.12,10.34,9.99,9.58]
 SET fastest FROM allTimes
 FOR EACH time FROM allTimes DO
    IF fastest TO time
       SET fastest to time
    END IF
END FOR EACH
SEND  "The winner time was" fastest TO DISPLAY
```

4.Write a pseudocode that reads in the three numbers and writes them all in sorted order

```
Read num1, num2, num3
If (num1 < num2)
   If(num2 < num3)
      Write num1 , num2, num3
```

```
    Else
        If(num3 < num1)
            Write num3, num1, num2
        Else
            Write num1, num3, num2
else
    If(num1 < num3)
        Write num2 , num1, num3
    Else
        If(num3 < num2)
            Write num3, num2, num1
        Else
            Write num2, num3, num1
```

5.Write a pseudocode that will count all the even numbers up to a user defined stopping point.
For example, say we want to see the first 5 even numbers starting from 0.
well, we know that evens numbers are 0, 2, 4, etc.
The first 5 even numbers are 0, 2, 4, 6, 8.
The first 8 even numbers are 0, 2, 4, 6, 8 ,10 ,12, 16

```
Read count
Set x to 0;
While(x < count)
    Set even to even + 2
    x = x + 1
    write even
```

Appendix M
<u>Questionnaire Survey For MSSNP Usability</u>

This survey is part of a study to evaluate the usability of Metacognitive Support System for Novice Programmer. You were specially selected as part of a random representative sample to take this survey. By responding to the survey, you will provide essential information about your perception on the usability factor of MSSNP. Please answer all questions

Metacognitive Support System for Novice Programmer (MSSNP) is a tool developed to support novice programming for learning programming metacognitively. Metacognition refers to a learner's ability automatic awareness of their own knowledge and their ability to understand, control, and manipulate their own cognitive process (Flavell, 1979). This role of metacognition in learning programming is one important in determining learning success. Five stages of metacognitive activities have been identified as being useful in supporting learners in learning computer programming as follows:

1. *Pre-Task – The main objective of this stage is for student to be stimulated by their own reflection in monitoring their knowledge. It focuses on the past experience in solving problems as well as the performance*
2. *Familiarization – The main objective of this stage is to encourage students to reflect on the strategies that may help them in problem-solving process*
3. *Production – The main objective of this stage is to reflect the student's understanding concerning the concept as well as their confidence in solving the problem correctly*
4. *Evaluation- This stage only involves the activity of checking the solution provided by the lecturer, which is used as a comparison in studying the student's solution.*
5. *Post-Task - The opportunity will be given to the students to review their most recent experiences as well as explore things that happened during the activity of solving the problem.*

Thank you for participating in this study!

Yours Sincerely,
Siti Nurulain Mohd Rum (Mrs.)
PhD Candidate
Faculty of Computer Science and Information Technology
University of Malaya
snurulain@siswa.um.edu.my

No.		Rating Score
Affect		
AF1	Working with this software is satisfying	
AF2	This software is awkward when I want to do something, which is not standard.	
AF3	Sometimes the MSSNP environment gives me a headache	
AF4	There are too many steps required to get something to work	
AF5	Sometimes MSSNP behaves in a way that I don’t understand	
AF6	I enjoy my sessions with MSSNP	
AF7	I would recommend this learning environment to my friends	
AF8	Working with this learning environment is mentally stimulating	
Efficiency		Rating Score
E1	The MSSNP has helped me to overcome any problems I have had by using it	
E2	The MSSNP is attractively presented	
E3	Using MSSNP reduces the time I spend in finding the right information for my early stage of research work	
E4	MSSNP allows me to find the information that I need for my research more quickly	
E5	Using MSSNP save my time	
Helpfulness		Rating Score
H1	The instructions and prompts are helpful	
H2	The way that information is presented is clear and understandable	
H3	The MSSNP instructions are very informative	
H4	There is enough information on the screen when it is needed	
H5	I can understand and act based on the information provided by MSSNP	
Control		Rating Score
C1	MSSNP responds too slow to inputs	
C2	I think MSSNP is inconsistent	
C3	The speed of MSSNP is appropriate	
C4	It is relatively easy to move from one part of a task to another	
C5	I can see at a glance what are the options at each stage	
Learnability		**Rating Score**
L1	It is hard to use new functions	
L2	The help information given by MSSNP is very useful	
L3	I’m sure in choosing the right command to use	
L4	It is easy to make MSSNP do exactly what I want	
L5	Learning to operate MSSNP is full of problems initially	

Do you have any other comments on the MSSNP system?

____

____

Appendix N
Experiment Materials

This appendix provides the list of all the materials used in the experiment with MSSNP

List of Materials

The materials used in this research work are presented in the following order:

1. Consent Form
2. Participation Information
3. Pre-test session for Part 1
4. Pre-test session for Part 2
5. Observation Form.
6. MSSNP manual (step-by-step tutorial).
7. Problem Solution workout sheet.
8. Post-test for Part 1
9. Post-test for Part 2

Appendix O

Observation Sheet (For Experimenter Only)

Participant No:______________________

Pre-Test result

(Prediction) / (Performance)	Unable to Solve it	Able to Solve it partially	Able to fully solve it
Provides correct answer			
Provides partial correct answer			
Provides Incorrect answer			

Post-Test result

(Prediction) / (Performance)	Unable to Solve it	Able to Solve it partially	Able to fully solve it
Provides correct answer			
Provides partial correct answer			
Provides Incorrect answer			