Metacognitive Support Accelerates Computer Assisted Learning for Novice Programmers

Siti Nurulain Mohd Rum[1*] and Maizatul Akmar Ismail[2]

[1]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Selangor, Malaysia // [2]Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia // snurulain@upm.edu.my // maizatul@um.edu.my

[*]Corresponding author

(Submitted February 29, 2016; Revised May 20, 2016; Accepted June 16, 2016)

ABSTRACT

Computer programming is a part of the curriculum in computer science education, and high drop rates for this subject are a universal problem. Development of metacognitive skills, including the conceptual framework provided by socio-cognitive theories that afford reflective thinking, such as actively monitoring, evaluating, and modifying one's thinking, has been identified as important for novice programmers. Studies have shown that metacognitive skills can be nurtured through the use of technology blended into educational activities. Designing metacognitive-related activities that focus on both social and cognitive development is both theoretically and practically challenging, especially in supporting the teaching and learning of computer programming. This paper describes six commonly-used strategies, viz., metacognitive scaffolding, reflective prompts, self-assessment, self-questioning, self-directed learning and graphic organizers, identified as important features that can be incorporated into computer-assisted learning tools in supporting computer programming learning. An experimental study was conducted to determine the effectiveness of these strategies. The results show that they helped learners by improving their performance in learning computer programming.

Keywords

Metacognitive, Support system, Self-regulation, Reflective prompts, Self-assessment, Self-questioning, Computer programming

Introduction

Interactive online educational technologies such as intelligent tutoring systems (ITS) can provide a good platform for performing research related to metacognition. Assessment of fine-grained tracking of students' cognitive abilities and their metacognitive behaviors can be provided by such systems. Designing a metacognitive support system focused on both cognitive and metacognitive development can be very challenging. The term metacognitive is most often associated with (Flavell, 1979) who has defined metacognitive knowledge as the process of acquiring knowledge about cognition to control cognitive processes. The role of metacognition in learning to solve computer programming problems is very important. Metacognitive management strategies are more often used by outstanding programming students than by lower-performing students (Bergin, Reilly, & Traynor, 2005). In fact, the more complex a programming problem, the greater the need for metacognitive control, purposeful reflection, and positive feedback (Havenga, 2011). A programmer must apply in-depth reading skills and meta-comprehension to judge how clearly and effectively he or she understands a programming problem. Furthermore, programmers must be skilled in problem-solving processes, apply appropriate programming approaches, and be able to correct programming errors, as well being able to think deeply about their programming solutions and test program output. Such problem-solving steps require metacognitive control such as planning (plan the solution), monitoring (monitor the design and development of the program) and evaluation (test and reflect on the programming solution). A well-trained programmer is someone equipped with good analytical thinking and problem-solving skills (Soloway & Spohrer, 2013). Students should therefore manage their skill with respect to programming processes, motivate their decisions, articulate their actions, and investigate alternative solutions to improve the quality of their programs. The teacher has a responsibility to support students in developing metacognitive skills and applying them during program development. Although there are methods of instruction (i.e., cognitive approach, motivation approach) the most valuable and effective methods involve a combination of theory and practice (Li, Zhang, Du, Zhu, & Li, 2015). The underlying strategies and knowledge of cognitive processes must be given to a learner along with opportunities to practice and apply both metacognitive and cognitive strategies. For development of metacognitive regulation, it is also important to evaluate the outcome of their efforts (Wegener, Silva, Petty, & Garcia-Marques, 2012). Yet, to date, there are many attempts have been made to provide the support learning tool for novice to learn computer programming (Sorva, Karavirta, & Malmi, 2013; Verdú et al., 2012). However, finding similar works that specifically discuss on improving novices' metacognitive skills using support learning tool could not be located. In the next section, we discuss the characteristics of a metacognitive support system

for learning computer programming; they are identified as scaffolding, reflective prompts, self-assessment, self-questioning, self-directed strategies and graphical organizer.

Metacognitive scaffolding

Difficulties encountered while learning computer programming are a universal problem. There have been numerous attempts to address these difficulties (e.g., (Apiola, Tedre, & Oroma, 2011; M. Rum, Nurulain, & Ismail, 2014; S. N. M. Rum & Ismail, 2014; Soloway & Spohrer, 2013), but challenges still remain, so an optimal support mechanism using learner and effective instructional strategies should be developed to provide an optimal learning environment for learning computer programming. These difficulties show that some programming skills required by novice learners may be beyond their capabilities. Scaffolding is a critical component in facilitating students' aptitude for programming (Bickhard, 2013; Feyzi-Behnagh et al., 2014). Scaffolding entails providing students with assistance that would increase their competence on an as-needed basis (Kim & Hannafin, 2011). With scaffolding, learners can engage in activities that otherwise would be beyond their abilities. Scaffolding refers to a variety of instructional techniques used by computer tutors and humans as well as such pedagogical agents as tools, guides and strategies in helping learners to develop understandings and competency that might otherwise be beyond their grasp (D'mello & Graesser, 2012). Learning difficulties can be reduced by providing such support outside the classroom, and it should augment, not replace, classroom learning. Several studies have shown that students often fail to gain basic understanding and exhibit poor performance when they use a computer-based learning environment to learn about complex topics without scaffolding (Azevedo, Cromley, Winters, Moos, & Greene, 2005; Greene & Land, 2000). Researchers have therefore begun emphasizing the significance of procedural, metacognitive, embedded, strategic, and conceptual scaffolds in computer-assisted learning environments. A review by (Jumaat & Tasir, 2014) has categorized scaffolding used in online learning into four main types; (1) conceptual scaffolding, (2) procedural scaffolding, (3) strategic scaffolding and (4) metacognitive scaffolding. Conceptual scaffolding guide learners to the key concepts of learning, procedural learning helps learners use appropriate resources as well as tools effectively; strategic scaffolding helps learners find appropriate strategies and methods in solving complex problems and metacognitive scaffolding helps learners to be prompted and reflected about what they are learning throughout the learning process. Metacognitive scaffolding not only promotes higher order thinking of learners but it also support learners ability to plan ahead.

Reflective prompts

Problem-solving is a common activity used as a teaching and learning approach in computer programming education because it helps students to develop different cognitive abilities. Solving problems requires a great deal of cognitive effort in activities such as finding solutions, identifying problems, and testing hypotheses. Prompting using self-regulated support and learning through problem solving is recognized as a valid instructional approach, especially in computer-assisted learning fields where simple prompting can be easily implemented (Bannert & Reimann, 2009, 2012). Research specifically focusing on self-regulated learning has shown that learners face many difficulties in performing self-regulation activities, and they may process information spontaneously, often failing to achieve desired learning outcomes (Bannert, 2006; Bannert & Reimann, 2012). Prompting students while engaged in task performance is an instructional strategy that seems promising for carrying out specific regulated activities. This method can be employed for supporting and guiding a learner in being self-regulated during the problem-solving process (Bannert & Reimann, 2012). Question prompts have been found by (Davis, 2000; Kim & Hannafin, 2011; Lai, 2008) to be an effective approach for eliciting reflection and helping student focus their attention as well as to monitor their learning through elaboration on the questions asked. To strength the quality of reflection, (J. A. Moon, 2004) proposed structuring reflection with questions. By performing reflective activities, learners would be able to identify the importance of various activities and thereby be able to develop understanding in a larger context such as responding to question prompts (Amulya, 2004). In a peer-tutoring context, recent research has shown evidence of a positive influence of an instructor's question prompts on a learner's reflective learning. Research shows that an instructor's question can motivate learner metacognition that could positively influence learning opportunities (Wu & Looi, 2011). The importance of incorporating question prompts into Intelligent Learning Environments (ILE) designs have also been recognized by researchers in ILE (e.g., (Murray et al., 2013; Van der Meij & de Jong, 2011). Question prompts have been used to scaffold learners towards appropriate learning and positive evidence of effectiveness of this approach in helping learners have been found, for example, in application to ill-structured problem-solving (Xie & Bradshaw, 2008) and integration of knowledge (Davis, 2000). Students can reflect on their own thoughts by motivating and activating them to analyze and think about the effectiveness of

strategies chosen to help them oversee, regulate, and control the application of procedures in a specific situation. Whether to display the question prompt in ILE usually depends on the purpose and intention of a specific interposition. A learner should receive the prompt from the learning tool at an appropriate time, e.g., at the moment where assistance is required; presenting a prompt in an inappropriate manner or at an inappropriate time will only cause cognitive overload (Ifenthaler, 2012; Thillmann, Künsting, Wirth, & Leutner, 2009). Generally, the assistance would be provided during, before, or after the learning sequence presentation. The presentation of a prompt during the sequence of learning is reasonable if the intention is to activate a learner's monitoring skill in problem-solving activities, but if the objective is to motivate the learner to evaluate certain problem-solving activities, then presentation after the learning sequence would be better. If one wishes to trigger learner reflection on determining an approach to solving a problem it would be appropriate to present a prompt before presenting the problem-solving sequence. Another crucial aspect is how a prompt can best be embedded to provide the learner with an optimal scaffold. There are two categories of reflective prompts, generic and directed; a generic prompt would seem more effective than a directed prompt because it gives the learner autonomy (Davis, 2003). On the other hand, a directed prompt would generally helpful for novices who are lack of problem-solving skills (Davis, 2003).

Self-assessment

One issue in teaching computer programming is student motivation. The learning process can be both effective and meaningful if students are given an opportunity to understand objectives for achieving goals and following their own learning processes. Focus on learning outcomes has been dominant at many universities (J. Moon, 2002). Learning outcomes refer to the knowledge level of learners in terms of what they know and are expected to do as a result of learning activity so this information can be used for goal-setting (Anderson et al., 2001). Self-assessment is a common method for assessment. Research has revealed that students perform better if they are able to evaluate their own work or the work of others (Anderson et al., 2001; Cassidy, 2006). Students who oversee and accurately evaluate their own performance may respond with appropriate study strategies to achieve their goal, so to engage them in activities that will encourage them to evaluate themselves and be able to explicitly explain what they know and do not know. Self-assessment provides potential for novice learners to improve their understanding.

Self-questioning

Self-questioning is an aid to metacognition, it can describe a process through which learners ask and answer questions while reading. This process enables them to understand and digest text information and become independent learners who can actively engage with organized thinking through goal-directed learning. Researchers in the metacognition field suggest that engaging in self-questioning is part of attributes of skilled reading (Williamson, 1996). Self-questioning is a type of skills that relates to self-management strategy that can be utilized to change behaviors, guide instruction, gain understanding, complete tasks, and many more (Joseph, Alber-Morgan, Cullen, & Rouse, 2016). In many cases, difficulties in comprehension are always related to failure of readers to actively participate in the reading process. To increase novice programmers' understanding of the important information from their reading sources as well to increase their motivation, self-questioning is one of the best approaches in practice. (M. Rum et al., 2014), for example, had designed reflective activities in his computer-assisted learning system to encourage self-questioning about learning experiences leading to self-directed learning. The more students used self-questioning in various kinds of situations, the more likely they were to develop self-questioning habits; this ultimately became an automatically-triggered skill used subconsciously as required in a given situation.

Self-directed learning

Self-directed learning (SDL) has received increased attention, particularly in the higher education context. (Malcolm Shepherd Knowles, 1975) advanced the most common foundation definition of SDL as a process in which individuals take the initiative without external help to manage their own learning, identify learning needs, formulate learning goals, implement appropriate learning strategies, apply required resources, and evaluate learning outcomes. The widely-known benefits of SDL are, traits of people who take initiative and responsibility in learning and learn more and better than those who do not (Malcolm Shepherd Knowles, 1975). (Gureckis & Markant, 2012) pointed out that SDL helps learners optimize their educational experience, allowing them to focus on useful information. The active nature of SDL also helps learners in the process of encoding information

and retaining it over time. Other benefits of SDL have been espoused by a number of theorists in adult education (Bolhuis, 2003; Malcolm S Knowles, Holton III, & Swanson, 2014). SDL is related to relationships between learners, self-directed learning, and the impact of technology (e-learning, internet and broadband) on self-directed learning (Hiemstra, 1994). In this research work, as suggested by (Mishra, Fahnoe, & Henriksen, 2013), the fact that the learners take initiative and responsibility for learning process, allowed them to manage, select and assess their own learning activities can be touted as self-directed learning.

Graphic organizers

Contemporary technologies appear to provide many opportunities for addressing the learning needs of novice programmers with respect to difficulty in learning computer programming concepts. (Fouh, Akbar, & Shaffer, 2012) suggest that the practice of using graphic representation of an algorithm provides a logical abstract for aiding learners in developing the logical thinking required in computer science courses. Figure 1 is an example of graphical representation of a data structure overview as an example of fundamental learning in computer programming.

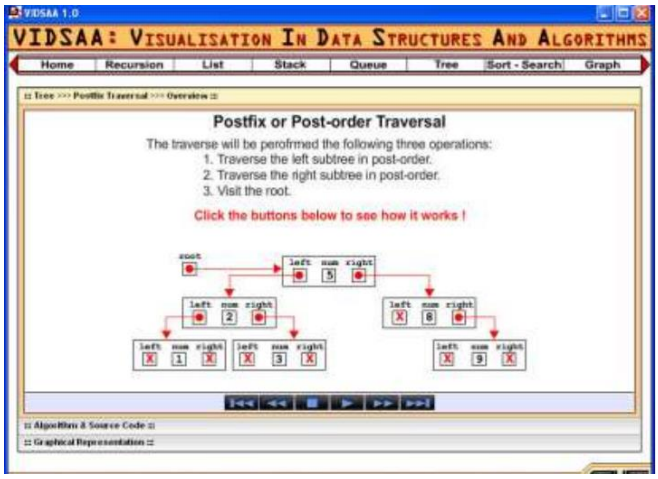

Figure 1. Graphical representation for data structure overview (Chansilp & Oliver, 2006)

Graphic representations have long been used to solve variety of problems and textual understanding. This technique can help learners analyze text by representing the structure and flow of textual information. Flowcharts, Venn diagrams, concept maps, and tree diagrams are examples of graphic organizers that can be used to better understand textual information. Network tree, cycles, fishbone maps and series and continua/scales are some identified graphic organizers useful in text-reading activity. A graphic organizer definition and common types of graphic organizers are presented in Figure 2.

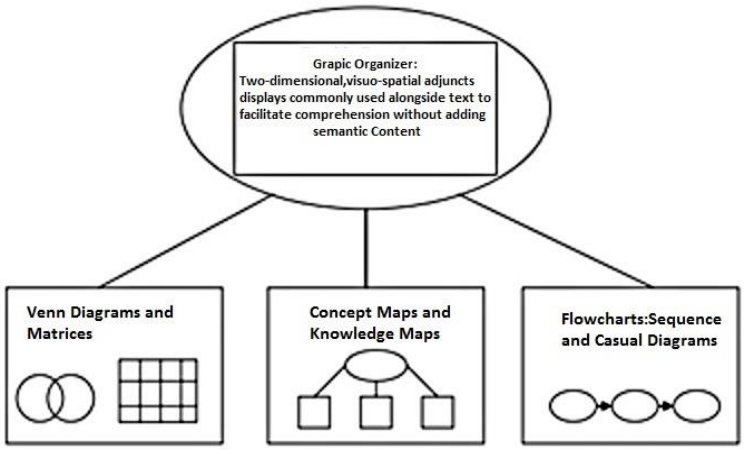

Figure 2. Graphic Organizer definition and common graphic organizer types (Otto & Everett, 2013)

Relationships between text elements and conceptual text structures can be illustrated using graphic organizers (Stull & Mayer, 2007). A graphic organizer is a text adjunct that is "a meaningful diagram formed from statements and words that graphically-connected is called a visuospatial information arrangement" (Horton,

Lovitt, & Bergerud, 1990). Concept maps, knowledge maps, Venn diagrams, causal diagrams, and matrices are just a few of the common graphic organizers found in textbooks (Stull & Mayer, 2007) and utilized in the classroom. When attempting to comprehend and learn from expository text, readers must be able to employ both surface-level and deep strategies. Surface-level strategies are those that help readers understand and remember fact-level text-based and macro-level information (Kintsch, 2005). Deep processing strategies help readers select important information, link new information to prior knowledge, and ultimately construct a situation or mental model (Kintsch, 2005). Graphic organizers help readers to achieve efficient learning and facilitate the process of knowledge construction through explicit connection with text concepts (Horton et al., 1990). Graphic organizers are particularly applicable to metacognitive reading strategy instruction and self-regulated learning because they can be utilized before, during, and after reading to monitor metacognitive and assess their learning.

Experimental studies

A system called Metacognitive Support Environment was developed to measure the effectiveness of proposed application support features. The motivation behind the development of this system was desire of novice programmers for an environment supporting learning computer programming metacognitively. The system support features as discussed in previous sections, i.e., scaffolding (support tailored on student's needs), self-questioning (encouraging the process of asking and answering while learning), self-assessment (the process of having the learners critically reflect upon and record the progress of their own learning), graphic organizer (used to organize information during learning) and self-directed learning (learner take the initiative without external help to manage their own learning) and reflective prompt (provides directive help for learner). The system is comprised of the following five main activities.

Pre-Stage (reflective stage, i.e., what learner's know and don't know) - The objective is to make students reflect both on what they know and do not know; this activity takes place before the process of learning begins. It presents appropriate conditions to make a student realize the importance of using suitable and possible strategies, providing reference resources as well as the degree of focus required for success in solving a problem. The main objective of this stage is for students to be triggered by their own reflections to monitor their knowledge. It focuses on their past experience in solving problems as well as the performance level indicated by low, average, or high. Students will be able to compare their estimation and judgment of their own actually-understood knowledge through exposure to activities such as knowledge monitoring and performance comparison as well as analysis of knowledge monitoring. These activities stimulate and encourage self-directed learning in which students are forced to take initiative and responsibility for learning.

Familiarization stage (assess learner's understanding) –the stage in which students both assess their understanding and develop a strategy for problem-solving. During this stage, students are presented with a list of possible strategies as most appropriate for solving the problem; alternatively they themselves can compose new strategies. The primary objective of this stage is to make them reflect on strategies that may help them solve the problem. This activity emphasizes metacognitive strategies that relate to the process of solving the problem. Students will be exposed to activities that allow them to compare their own knowledge judgment with their actual knowledge and help them select the strategies provided for solving the problem. One type of metacognitive support element provided during this stage is the reflective prompt used to provide directive help for learner in the learning process. Figure 3 presents an example of reflective prompts and graphical organizer used to parse postfix expression into an expression tree. In this learning example, learners were asked to build an expression tree based on the given postfix expression. The operations buttons were provided for the learner to choose to structure the expression in the graphical form.

Production stage (learner's self-assessment and problem comprehension) is the stage in which students perform self-assessment and deal with problem comprehension. In this stage, students are required to solve the given problem by presenting an answer. The main objective in this stage is to reflect an understanding with respect to the concept as well as demonstrating confidence in correctly solving the problem. This activity relates to the student's performance self-assessment. Activities taking place during this stage concentrate on translating the given problem into pseudocode and monitoring the application of the planned strategies. The activities are comprised of problem-solving, checking answers, and quizzes. Students are thereby exposed to self-assessment activity where they must evaluate their own work.

Evaluation stage (learner's experience of past problem solving) - the stage in which students evaluate their own experience in past problem solving. This stage involves only the activity of checking the solution provided by the lecturer, used as a comparison for studying the student solution. Graphical representation would be used to

convey a learner's metacognition information in the form of reflectometers to trigger knowledge monitoring accuracy and time management (see Figure 3).

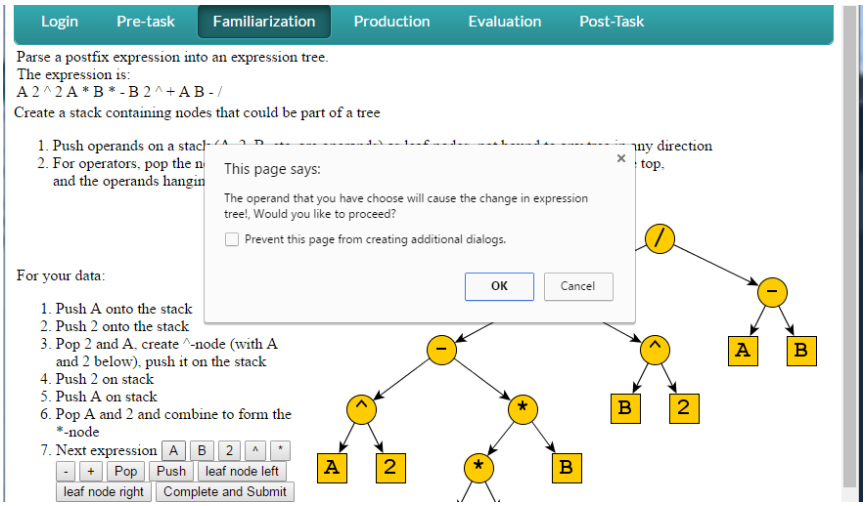

Figure 3. Reflective prompts and Graphical organizer used to convert postfix expression into an expression tree

Post-task stage (learner's to reflect on the problem solved) –The main objective of this stage is to have the students reflect on the problem solved by providing an opportunity to review their most recent experiences as well as to explore things that happened during the activity of solving the problem. Students should be able to identify the "cause of the mistakes" that occur during the problem-solving activity, the time spent, and the resources used. As suggested by (Quintana, Zhang, & Krajcik, 2005) in their framework for scaffolding tools, a progress bar is provided during this stage to enable students to see the their most recent activity timelines.

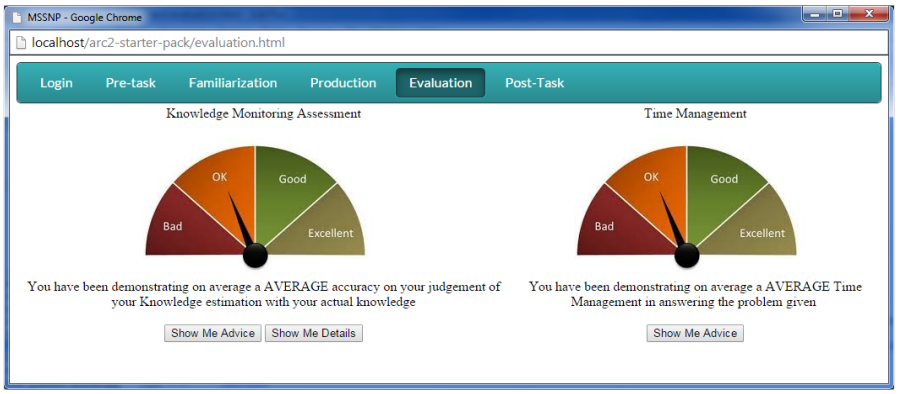

Figure 4. Evaluation dashboard screenshot

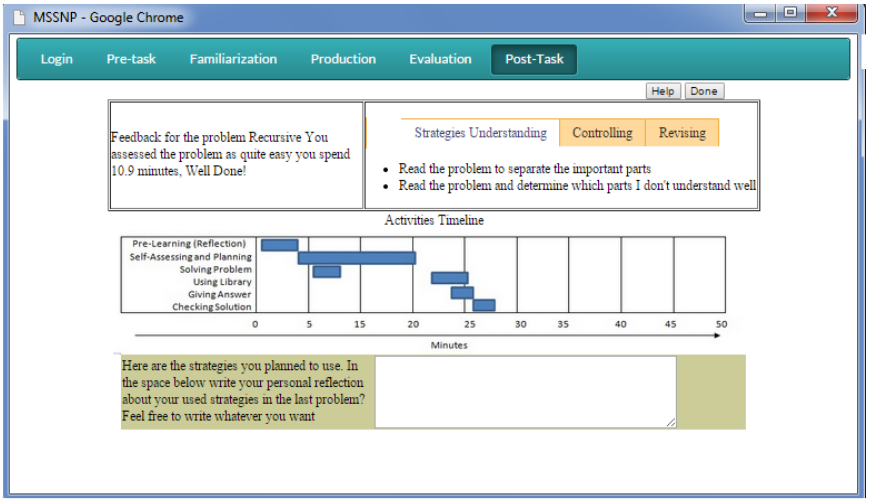

Figure 5. Post-task screenshot

Figure 4 is a screenshot of evaluation dashboard, and Figure 5 is a screenshot of post-task activity; these are elements of the proposed system.

A discussion of development of the proposed support system can be found in (M. Rum et al., 2014; S. N. M. Rum & Ismail, 2014). The primary target of the proposed system is to scaffold the knowledge monitoring skill of a programmer, i.e., to provide the ability and opportunity to assess one's knowledge, or by extension, one's understanding. We believe that promoting awareness of knowledge monitoring accuracy at the novice level is the first step in fostering metacognitive skills that in turn help in making a good selection of strategies and facilities for better allocation of cognitive resources.

Participants

An experimental study was conducted with two different groups of subjects: a control group and an experimental group each associated with different conditions. A four-week training course, Introductory Computer Programming, was given to each group. Throughout the training sessions, only the experimental group was exposed to and interacted with the proposed support system, while respondents in the control group were given only ordinary class sessions without interaction with the proposed support tool. Sixty-six of 100 target respondents (first year Computer Science undergraduate students) were invited via email and agreed to take part in this study. They were randomly assigned into two groups namely experimental and control group. From the total of 66 participants, 30 agreed to take part in the experimental group and 36 in the control group. The distribution of respondents is shown in Table 1.

Table 1. Distribution of respondents by group

Group	Frequency	Percentage
Experimental	30 (Male = 12, Female = 18)	45
Control	36 (Male = 15, Female = 21)	55
Total	66	100

Method with research design

The experimental design had a pre-test and post-test similar to the underlying idea for evaluating the (Tobias, Everson, & Laitusis, 1999) model of metacognition. These tests were designed to measure basic knowledge concepts related to programming and knowledge monitoring. A pre-test was given to both groups before the training session and a post-test given for both groups after the training session. The tests (both pre- and post-test) were divided into two parts. In the first part, for both tests, 10 minutes was given to the students to complete the tests; 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 were asked to answer "yes" or "no" to the question: "Do you think you can translate the given problem into pseudocode?" In the second part, again for both pre- and post-test, they were required to translate the same problems into pseudocode. All participants (pre- and post-test) were given 1 hour to complete these tests. The post-test was designed to be more difficult than the pre-test, given that the subjects should have gained more practice in solving problems during their training. Both tests (pre-test and post-test) had five questions in total and were devised with the assistance of a computer programming instructor. Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is "text-based" (algorithmic) and written in English as a detailed yet readable description of what a computer program or algorithm must do, expressed in a formally-styled natural language rather than in a programming language. Pseudocode makes creating programs easier and allows programmers to concentrate on the logic of the problem to be solved (the algorithm) without having to know programming language syntax in detail. It depicts the entire algorithm's logic so that it requires only rote implementation to translate it line by line into executable source code.

Procedure and measurement instruments

The scoring system was as follows: 2.0 points for a correct answer (the logic of the written pseudocode is totally correct), 1.0 point for a partially correct answer (the logic of the written pseudocode is partially correct), 0.5 points for an incomplete answer (the logic of the written pseudocode is incomplete) and 0 for an incorrect answer (the logic of the written pseudocode is totally incorrect). Scoring was performed by computer programming experts, i.e., faculty lecturers, each with more than 5 years of experience in teaching computer

programming. For measuring the knowledge monitoring assessment of learner (designated KMA). This instrument have been used by many researchers (Moran, 2012; Sangin, Molinari, Nüssli, & Dillenbourg, 2011) to evaluate student knowledge monitoring ability across domains. We adapted the empirically-validated instrument developed by (Tobias & Everson, 1996) after changing it slightly by giving a little additional flexibility to the possibility that a student predicted that they would partially solve the problem or that they partially understood it. Table 2 presents the knowledge monitoring assessment results with score values of a, b, c, d, e, f, g, h, i. The KMA exhibited nine possible scores reflecting the relationship between a student's prediction of their knowledge and her actual performance. A score of 1 was given for the situations a, e and i, while a score of -1 was given for situations c and g, and a score of -0.5 was given for situations b, d, f, and h. The mean of knowledge monitoring scores over all the problems solved represents the current KMA state of the Student.

Table 2. KMA Score Condition

Actual performance	Unable to solve it	Able to solve it partially	Able to solve it
Provides incorrect answer	a	b	c
Provides partial correct answer	d	e	f
Provides correct answer	g	h	i

Results

Wilcoxon tests were performed to investigate the repeated measures of difference between the groups. In this study, if the research hypothesis is true, it is expected that positive rank is higher in experimental group and similar numbers for both positive and negative rank in control group. An examination of the findings shown in Table 3 reveals that there was a significant difference between the pre-test and post-test scores of students in the experimental group (Z = -2.292, p = .000 < .001). The sum of negative ranks for the experimental group students' academic achievement scores was 7.50, while the sum of positive ranks was 58.5. There were 20 respondents show an increase in terms of performance improvement in learning Introductory Computer Programming before and after training session, only 2 of respondents from experimental show a drop on performance and 8 respondents were not affected with the four-week training with the aid of proposed system throughout the training session. Given the sum of ranks for the difference scores, the result shows that there was a significant improvement in post-test scores as compared to pre-test scores obtained by the experimental group.

Table 3. Wilcoxon test results for pre- and post-test scores changes of experimental group

Ranks		*N*	Mean rank	Sum of ranks	*Z*	*p*
PostTest - PreTestExp	Negative Ranks	2[a]	7.50	7.50	-2.292	.022
	Positive Ranks	20[b]	5.85	58.50		
	Ties	8[c]				
	Total	30				

Note. [a]PostTest < PreTestExp; [b]PostTest > PreTestExp; [c]PostTest = PreTestExp.

The results in Table 4 show that there was no difference between the pre-test and post-test scores of the control group, where the sum of the negative ranks of the control group was 2.5 and the sum of the positive ranks was 1 with Z = -1.089, p = .0276 at the 5% level. There were 30 respondents in this group did affected with the given training session, 4 respondents did not show a drop in performance before and after the training session, only 2 respondents show positive on the improvement of performance in learning Introductory Computer Programming. Given the sum of ranks for the difference scores of the control group, the result shows that there was no difference of performance changes in control group for the four weeks training.

Table 4. Wilcoxon test for pre- and post-test score changes of control group

Ranks		*N*	Mean rank	Sum of ranks	*Z*	*p*
PostTest - PreTestExp	Negative Ranks	4[a]	2.50	5.00	-1.089	.276
	Positive Ranks	2[b]	1.00	1.00		
	Ties	30[c]				
	Total	36				

Discussion

Research into the challenges of teaching and learning programming to novices has long received attention to many researchers especially when dealing with conceptually rich domains that require metacognitive perspective

as a vehicle to stimulate the learning process. In this study, we proposed a metacognitive educational environment for learning Introductory Computer programming for novice programmers. We have restricted our definition of novices to tertiary level students who are learning programming for the first time. Parse problem solving in the form of pseudocode before translating into actual programming language is a common method in computer programming pedagogy. Respondents that participated in this study were novice programmers randomly selected and assigned into two groups; control group and experimental group. Both groups were received four weeks training on Introductory Computer Programming, however, only the experimental group had been exposed to and interacted with the proposed support system throughout the training session. In this study, we have developed the hypothesis that the more student interact with the proposed system the more reliable the proposed support features of the system. The KMA was used as the instrumental measurement of student performance in learning Introductory Computer Programming. In Table 3, the result of experimental group shows that 66.7% (20) of respondents show an improvement in post-test after received four-weeks training and exposed to and interacted with the proposed support system throughout the training session, while 26.7% (8) of the respondent did not show any improvement, meaning that there is no differences of performance changes between pre-test and post-test and only 6.6% (2) respondents show an inclination on the performance in learning Introductory Computer Programming subject. In Table 4, the result show that there was no differences of performance gained by the majority of respondents in control group (84%) (30) Before and after the training session, only 5% of the respondents show an increase in performance after received the training session, while 11% of the respondents show an inclination on the learning performance. From the result, it can be concluded there were positive effects in terms of performance gained in learning computer programming with the aid and interaction of the proposed support tool during the training session. This was might be due to the effect of metacognition support elements provided in the support tools. The finding of this research work support the hypothesis and consistent with the studies done by other researchers (Azevedo, Moos, Johnson, & Chauncey, 2010; Bernard & Bachu, 2015; Feng & Chen, 2014; Ismail, Ngah, & Umar, 2010). The result demonstrates that the proposed metacognitive strategies of computer-assisted learning support (metacognitive scaffolding, reflective prompts, self-assessment, self-questioning, self-directed learning and graphic organizer) helps learners in improving their learning performance in computer programming, especially in knowledge monitoring and problem-solving.

Conclusion and future study

In the present study, we have discussed characteristics that should be considered for developing computer-assisted learning tools in a metacognitive learning environment for supporting teaching and learning of computer programming. The present study support the hypothesis, that with the aid metacognitive strategies, student would be able to achieve their goal in learning computer programming, perhaps, in order to have better understanding the learning success is to view and measure all other factors such as learner motivation state, cognitive strategies and etc. This is the possibility of future investigation of how other factors contribute to learning success in computer programming subject. It is also important for more research to be conducted in this area with stronger controls in order to show whether or not metacognitive accuracy can be improved further. For the improvement of this research work, it is also suggested for future work to get more respondents to be participated as well as to increase the duration of training session (e.g., one semester) to see the effectiveness of the proposed system. We summarize the benefits of metacognitive strategies as discussed in this research work in Table 5.

Table 5. The benefits of the proposed support features

Features	Description	Benefits
Metacognitive scaffolding	Provides coaching, guides and advice	Strengthens the transferability of digital learning by helping to improve novice learner adaptability to a particular learning situation
Reflective prompts	Provides directive helps for learner during learning process	Encourage learners to reflect on techniques and strategies that they employ in learning process
Self-questioning	Learners ask and answer while learning	To aid learner's in comprehension monitoring strategy; it is very useful in assisting novices to become self-directed and independent.
Self-directed learning	Learners take initiative and responsibility in learning process	Learner more willing and motivated to learn because of the ability to manage, select and assess his/her own learning activities.
Graphic organizer	Graphical representation used to solve a variety of problems.	The complexity of a computer programming subject can be reduced by helping learners in developing the logical thinking required by computer science courses.

References

Amulya, J. (2004). *What is reflective practice.* Cambridge, MA: The Center for Reflective Community Practice at MIT. Retrieved from http://www.itslifejimbutnotasweknowit.org.uk/files/whatisreflectivepractice

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths, J., & Wittrock, M. C. (2001). *A Taxonomy for learning, teaching, and assessing: A Revision of Bloom's taxonomy of educational objectives, abridged edition.* White Plains, NY: Longman.

Apiola, M., Tedre, M., & Oroma, J. O. (2011). Improving programming education in Tanzania: Teachers' and students' perceptions. In *Frontiers in Education Conference (FIE)* (pp. 1-7). doi:10.1109/FIE.2011.6142787

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.

Azevedo, R., Moos, D. C., Johnson, A. M., & Chauncey, A. D. (2010). Measuring cognitive and metacognitive regulatory processes during hypermedia learning: Issues and challenges. *Educational Psychologist, 45*(4), 210-223.

Bannert, M. (2006). Effects of reflection prompts when learning with hypermedia. *Journal of Educational Computing Research, 35*(4), 359-375.

Bannert, M., & Reimann, P. (2009). Metakognitives Fördern des Lernens mit digitalen Medien durch Prompting-Maßnahmen1 [Metacognitives Promoting learning with digital media through prompting measures1]. *Lernchance Computer. Strategien für das Lernen mit digitalen Medienverbünden, 52*, 67.

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

Bergin, S., Reilly, R., & Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. In *Proceedings of the First International Workshop on Computing Education Research* (pp. 81-86). New York, NY: ACM.

Bernard, M., & Bachu, E. (2015). Enhancing the metacognitive skill of novice programmers through collaborative learning. In *Metacognition: Fundaments, applications, and trends* (pp. 277-298). doi:10.1007/978-3-319-11062-2_11

Bickhard, M. H. (2013). Scaffolding and self scaffolding: Central aspects of development. In L. T. Winegar & J. Valsiner (Eds.), *Children's Development within Social Context* (pp. 33-52). Hillsdale, NJ: Lawrence Erlbaum Associates.

Bolhuis, S. (2003). Towards process-oriented teaching for self-directed lifelong learning: A Multidimensional perspective. *Learning and instruction, 13*(3), 327-347.

Cassidy, S. (2006). Learning style and student self-assessment skill. *Education+ Training, 48*(2/3), 170-177.

Chansilp, K., & Oliver, R. (2006). Reusable and shareable learning objects supporting students learning of data structures in university courses. In Proceedings of International Conference of EDU.COM (pp. 105-113). Nong Khai, Thailand: Edith Cowan University.

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. (2003). Prompting middle school science students for productive reflection: Generic and directed prompts. *The Journal of the Learning Sciences, 12*(1), 91-142.

D'Mello, S., & Graesser, A. (2012). AutoTutor and affective AutoTutor: Learning by talking with cognitively and emotionally intelligent computers that talk back. *ACM Transactions on Interactive Intelligent Systems (TiiS), 2*(4), 23. doi:10.1145/2395123.2395128

Feng, C. Y., & Chen, M. P. (2014). The Effects of goal specificity and scaffolding on programming performance and self-regulation in game design. *British Journal of Educational Technology, 45*(2), 285-302.

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

Fouh, E., Akbar, M., & Shaffer, C. A. (2012). The Role of visualization in computer science education. *Computers in the Schools, 29*(1-2), 95-117.

Greene, B. A., & Land, S. M. (2000). A Qualitative analysis of scaffolding use in a resource-based learning environment involvingthe world wide web. *Journal of Educational Computing Research, 23*(2), 151-179.

Gureckis, T. M., & Markant, D. B. (2012). Self-directed learning a cognitive and computational perspective. *Perspectives on Psychological Science, 7*(5), 464-481.

Havenga, M. (2011, July). *Problem-solving processes in computer programming: A Case study*. Paper presented at the Southern African Computer Lecturers' Association (SACLA), Ballito, South Africa.

Hiemstra, R. (1994). Self-directed learning. In *The sourcebook for self-directed learning* (pp. 9-20). Amherst, MA: HD Press.

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.

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. N., Ngah, N. A., & Umar, I. N. (2010). Instructional strategy in the teaching of computer programming: A Need assessment analyses. *The Turkish Online Journal of Educational Technology (TOJET), 9*(2).

Joseph, L. M., Alber-Morgan, S., Cullen, J., & Rouse, C. (2016). The Effects of self-questioning on reading comprehension: A Literature review. *Reading & Writing Quarterly, 32*(2), 152-173.

Jumaat, N. F., & Tasir, Z. (2014). Instructional scaffolding in online learning environment: A Meta-analysis. In *International Conference on Teaching and Learning in Computing and Engineering* (LaTiCE) (pp. 74-77). doi:10.1109/LaTiCE.2014.22

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.

Kintsch, E. (2005). Comprehension theory as a guide for the design of thoughtful questions. *Topics in Language Disorders, 25*(1), 51-64.

Knowles, M. S. (1975). *Self-directed learning: A Guide for learners and teachers*. Englewood Cliffs, NJ: Prentice Hall.

Knowles, M. S., Holton III, E. F., & Swanson, R. A. (2014*). The Adult learner: The Definitive classic in adult education and human resource development*. New York, NY: Routledge.

Lai, G. (2008). *Examining the effects of selected computer-based scaffolds on preservice teachers' levels of reflection as evidenced in their online journal writing* (Unpublished doctorial dissertation). Georgia State University, Atlanta, GA.

Li, J., Zhang, B., Du, H., Zhu, Z., & Li, Y. M. (2015). Metacognitive planning: Development and validation of an online measure. *Psychological assessment, 27*(1), 260.

Mishra, P., Fahnoe, C., & Henriksen, D. (2013). Creativity, self-directed learning and the architecture of technology rich environments. *TechTrends, 57*(1), 10-13. doi:10.1007/s11528-012-0623-z

Moon, J. (2002). *The Module and programme development handbook: A Practical guide to linking levels, learning outcomes & assessment.* Sterling, VA: Psychology Press.

Moon, J. A. (2004). *A Handbook of reflective and experiential learning: Theory and practice.* New York, NY: Psychology Press.

Moran, S. C. (2012). *The Effects of defensive pessimism and metacognitive bias on metacognitive knowledge monitoring and academic achievement* (Unpublished doctoral dissertation). Kent State University, Kent, OH.

Murray, T., Wing, L., Woolf, B., Wise, A., Wu, S., Clark, L., Xu, X. (2013). A Prototype facilitators dashboard: Assessing and visualizing dialogue quality in online deliberations for education and work. In *Proceedings of the International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE)* (pp. 36-42). Athens, Greece: The Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Otto, C. A., & Everett, S. A. (2013). An Instructional strategy to introduce pedagogical content knowledge using Venn diagrams. *Journal of Science Teacher Education, 24*(2), 391-403.

Quintana, C., Zhang, M., & Krajcik, J. (2005). A Framework for supporting metacognitive aspects of online inquiry through software-based scaffolding. *Educational Psychologist, 40*(4), 235-244.

Rum, M., Nurulain, S., & Ismail, M. A. (2014). *Ontology development of metacognitive support system for novice programmers (MSSNP)*. In *International Symposium on Technology Management and Emerging Technologies (ISTMET)* (pp. 316-321). doi:10.1109/ISTMET.2014.6936526

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). Retrieved from http://ajouronline.com/index.php/AJEEL/article/view/1819

Sangin, M., Molinari, G., Nüssli, M.-A., & Dillenbourg, P. (2011). Facilitating peer knowledge modeling: Effects of a knowledge awareness tool on collaborative learning outcomes and processes. *Computers in Human Behavior, 27*(3), 1059-1067.

Soloway, E., & Spohrer, J. C. (2013). *Studying the novice programmer.* New York, NY: Psychology Press.

Sorva, J., Karavirta, V., & Malmi, L. (2013). A Review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE), 13*(4), 15. doi:10.1145/2490822

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), 808-820.

Thillmann, H., Künsting, J., Wirth, J., & Leutner, D. (2009). Is it merely a question of "What" to prompt or also "When" to prompt? The Role of point of presentation time of prompts in self-regulated learning. *Zeitschrift für Pädagogische Psychologie, 23*(2), 105-115.

Tobias, S., & Everson, H. T. (1996). *Assessing metacognitive knowledge monitoring*. New York, NY: College Entrance Examination Board.

Tobias, S., Everson, H. T., & Laitusis, V. (1999). Towards a Performance Based Measure of Metacognitive Knowledge Monitoring: Relationships with Self-Reports and Behavior Ratings.

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.

Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P., & Queirós, R. (2012). A Distributed system for learning programming on-line. *Computers & Education, 58*(1), 1-10.

Wegener, D. T., Silva, P., Petty, R. E., & Garcia-Marques, T. (2012). The Metacognition of bias regulation. *Social Metacognition*, 81-99.

Williamson, R. A. (1996). Self-questioning-An aid to metacognition. *Reading Horizons, 37*, 30-47.

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), 1-7.

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.