DOES A COMBINATION OF METAPHOR AND PAIRING ACTIVITY HELP PROGRAMMING PERFORMANCE OF STUDENTS WITH DIFFERENT SELF-REGULATED LEARNING LEVEL?

Tie Hui Hui
Centre for Postgraduate Studies
SEGi College Penang, Malaysia
tiehuihui@gmail.com

Irfan Naufal Umar
Centre for Instructional Technology & Multimedia
University Science Malaysia, Malaysia
irfan@usm.my

ABSTRACT
This study aims to investigate the effects of metaphors and pairing activity on programming performance of students with different self-regulated-learning (SRL) level. A total of 84 computing students were involved in this seven-week study, and they were randomly assigned either to a group that received a combination of metaphor and pair programming (MPP) or to another group that received pair programming (PP) only. Students in both groups worked in pairs according to their SRL level (one high and one low) when solving programming problems in C++ language. The findings revealed that high SRL students in the MPP method performed significantly better in recall than their peers in the PP method, and similar result was observed among the low SRL students. However, no interaction effect was observed between the method and SRL level on programming performance, i.e., high SRL students always perform better either in the MPP or PP groups. Metaphors have assisted the learners to develop better conceptual understanding by linking the known to newly acquired abstracts; and pair programming does cultivate peer discussions. Also, instructor should assist students to improve their SRL to reinforce self learning.
Keywords: Metaphors, Self-Regulated Learning, Recall Performance, Pair Programming, Computer Programming

INTRODUCTION

Computer programming as part of the computing education is an essential skill that ought to be grasped by students in studying computer science. As programming demands complex cognitive skills, students find it difficult to understand, interpret and perform these complex cognitive tasks (Hawi, 2010; Mayer, 2003). Likewise, educators involved in teaching programming concepts to first year computing students are continually facing different challenges in cultivating the students' understanding in the fundamental area of semantics which is the programme comprehension. Miliszewska and Tan (2007) stated that complex cognitive skills such as planning, reasoning, problem solving and analytical thinking play their role in learning to programme. Problem solving skills which include reasoning and analytical thinking are required in analysing the given problem scenario. During the learning process, students are required to understand the given problem, design, code and perform maintenance that involve complex cognitive and social activity. To the first year computing students, majority of them believe that programming skill is complex and difficult to learn. However, those who are passionately interested in exploring the abstract problems find themselves motivated in acquiring the programming skill. Usually, these students are actively engaged in class activities and during lectures while the programming topics are covered. Somehow, they are able to seek help and discuss problems relating to programming. As such, effective learning takes place when students are learning through positive peer pressure in a fun and joyful environment as well as to reflect on self-learning outcomes by comparing them to the initial goals. Furthermore, higher thinking skill is needed in order for students to be the creators of new ideas, analyzers of information and generators of knowledge which seem lacking in these students (Butler & Morgan, 2007).

An earlier research on cultivating thinking and problem solving skills within students has been carried out when Pseudocode and program flowchart are mainly focused on the basic programming constructs (Tie, 2011). Besides teaching programming concepts, educators have tried in vain to cultivate the skills such as critical thinking, analytical and problem solving which are crucial to students who intent to take up programming career. Over emphasizing on the program syntax and semantics of individual statements will lead to the students' misunderstanding and inability to construct a complete working system which is the pragmatics. Despite the fact that students could recognise the syntax and semantics errors in the program flowchart or Pseudocode, they might not notice the logical errors. Foremost, these students find it a challenge when they were asked to convert the programming logic (in the program flowchart or Pseudocode) into executable programming codes in C++

language.

LITERATURE REVIEW

Metaphor is a high level abstract concept that involves the presentation of new idea in terms of relating it to the existing knowledge. American Heritage Dictionary Editors (2000) defined metaphor as a figure of speech in which the understanding of one thing is used to describe another. This is used to show that the two things are having the same qualities which making it an absolute comparison. It consists of two terminologies: the target and the source. As defined by Lakoff and Johnson (2003), the target is the subject to which attributes are assigned. The source is the subject from which attributes are borrowed, that is called to describe the target. Teaching approach attempted to cover numerous fundamental C++ concepts, for example variables declaration, data types, classes and control structure. Therefore, it is important that the technique focused on concepts which the students have seen before and build upon them. In this case, metaphor is used to communicate C++ concepts to students in a way that they could assimilate them and relate them to what they already know. With this, it is significant to assist the formation of interpretation and application of knowledge from the basic programming concepts acquired. Mastering the basic programming skills is fundamental for preparing learners to the next higher programming courses. Metaphors play a significant role in helping learners to develop mental images to reason abstract situations. They are being described as a real world system which the students are able to apply as a reference for linking existing ideas to the newly introduced concepts in programming system (Parker, 2009). The metaphor is expressed into either visual or textual representation in relating the abstract nature of the programming tasks to the fundamental of programming concepts. In learning programming syntax, educators use metaphor for communicating novel concepts. In turn, students identify the anomalies between their existing knowledge and the new information by the metaphor and develop new knowledge by connecting their existing knowledge to accommodate both sources (target and source). In this case, students are to transform these abstract concepts into logical flow by using designing tools such as program flowchart and Pseudocode before converting it into C++ programming codes. By connecting any concrete images with text information it will improve understanding in learning programming and increase the learners' recall (Flanik, 2008). Thus, metaphor as an instructional strategy used deliberately in communication to achieve specific effects that transform students' programming performance. It assists in enhancing programming comprehension and better academic performance. Three examples of conceptual metaphors (Figure 1, Figure 2 and Figure 3) are used to illustrate the C++ syntax in learning programming.

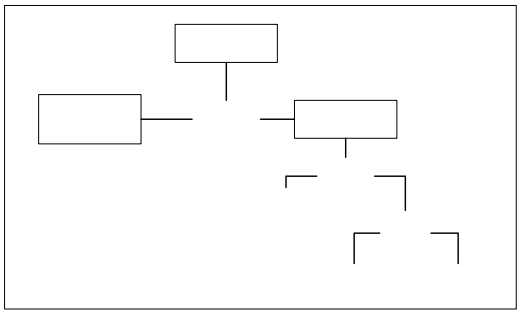

Figure 1: An Symbol Expression Tree Formed for Assignment (adopted from Merwe, 2008)

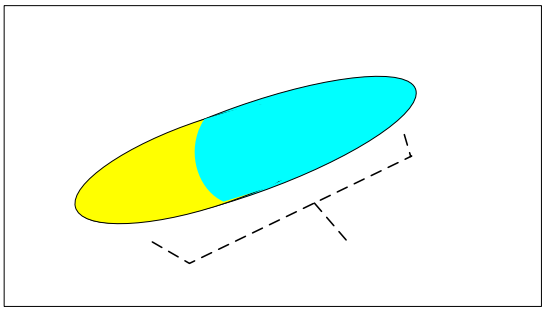

Figure 2: Medical Capsule Representing Classes

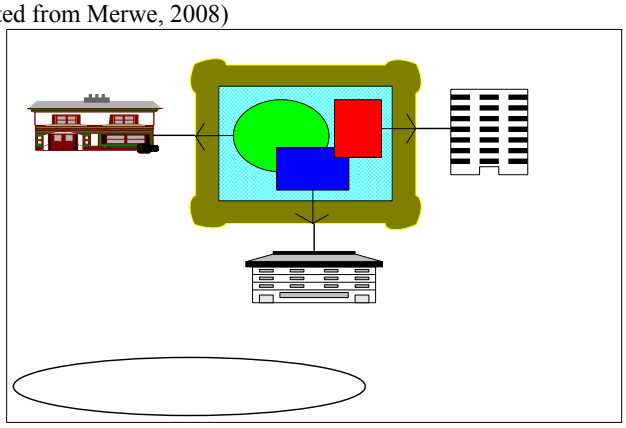

Figure 3: The Library Metaphor – explaining the "class" concept

Pair programming is a structural and systematic form of programming cooperation. It has been adopted in software industry to increase programmers' productivity and programming skills, where programmers work together in groups to complete the assigned tasks (Chung & Lo, 2006; Beck, 2000); and in education to increase learning. Research findings revealed that students perform better in terms of producing higher quality of codes, increasing retention rates, as well as improving problem solving skills and attitude towards programming when working in pairs (Bruce & McMahon, 2002).

Self-Regulated Learning (SRL) has been defined as a process in which the students set goals for their learning. It is a skill with the ability to regulate learning towards a desirable learning outcome. This includes planning and carrying out certain strategies for the achievement of the goals, and to independently manage time and effort, and evaluate the quality of their own learning environment (Jossberger, Brand-Gruwel & Boshuizen, 2006; Zimmerman, 2000). It also involves motivation, time management, behavior, physical and social environment regulation. Kerka (2005) indicated that the learning process of students and their performance are closely associated with the different levels of SRL abilities. He also revealed that SRL has positive effects on their learning abilities in terms of cognitive, attitudes, behaviours, emotional and psychological development, and personal empowerment. The students' level of SRL ability, high or low, is based on the group mean measured using the Motivated Strategies for Learning Questionnaire (MSLQ) instrument developed by Pintrich and DeGroot (1990). Studies have shown a significant correlation between an individual student with a high level programming performance and his high-quality involvement in SRL (Zimmerman, 2008; Lee, Shen & Tsai, 2008). In fact, the high SRL students are those who are highly involved in independent learning (Reyero & Touron, 2003). These students have the ability to regulate learning towards a desirable learning outcome and the skill to manage and organize their own learning needs, strategies and learning opportunities. The students with higher level of SRL ability are capable of building their own conceptual metaphors when new ideas are presented. By relating the existing knowledge and experiences to the newly introduced concepts, these highly self-regulated students are competent to set their learning goals based on own expected learning outcomes. When feedbacks and constructive criticisms are obtained from lectures, these learning strategies will be refined to ensure effective learning with positive outcomes.

RESEARCH QUESTIONS

In this study, three primary questions have been formulated to address the research outcomes:

RQ1: Is there any significant difference in terms of recall performance for high SRL students who received a combination of metaphor and pair programming (MPP) treatment and those who received only the pair programming (PP) method?

RQ2: Is there any significant difference in terms of recall performance for low SRL students who received MPP treatment and those who received only the PP method?

RQ3: Is there any interaction effect between instructional methods and self-regulated learning level?

RESEARCH METHODOLOGY

The purpose of this study is to investigate the effects of blending the metaphor with pair programming strategy on the programming recall performance among high and low SRL computing students in learning programming constructs through C++. It aims to examine whether the different levels of SRL could be the moderating factors when an instructional strategy such as (i) metaphors as visualisation technique, and (ii) pair programming as cooperative learning, are used in both classroom and practical sessions during course delivery.

Research Design

A 2 x 2 factorial design was applied to examine the effects of MPP and PP instructional methods on the students' recall performance. This quasi-experimental study applied pre and post-test control group design as illustrated in Figure 4. In this case, the self-regulated learning level (high and low) was used as the moderating variable. The students' recall performances were measured based on the immediate post-test scores obtained from the Computer Programming Performance Test (CPPT). All the 84 students (n = 84) from the first year semester one undergraduate computing course were involved in this study. These two classes, all intact groups, were randomly assigned to the two treatment groups. The experimental group (n= 42) received the MPP treatment while the control group (n= 42) was treated with the PP method. For this study, the course comprised lectures and practical / tutorial sessions. During the lecture session, the students were given the explanation on some programming concepts using tools such as flowcharts and Pseudocode, while during the tutorial or practical session, the students used the C++ language for coding. This was carried out for seven weeks on the two treatment groups in the classrooms with practical session where the pre-test was conducted before the treatment and the immediate post-test was conducted immediately after the treatment.

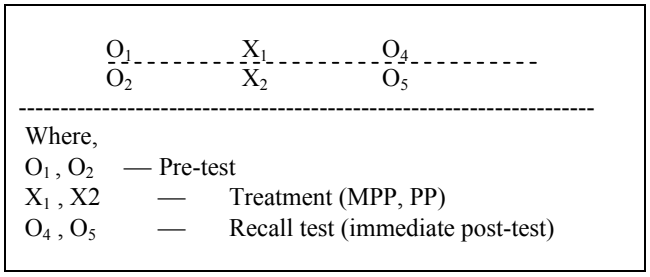

Figure 4: The Overall Research Design

Research Instruments

Prior to the study, the Motivated Strategies for Learning Questionnaire (MSLQ) was used to identify the students' self-regulated learning level. It consists of 23 items that requires 20 minutes to complete. In this study, the MSLQ mean score of the sample was 3.50. Students who scored 3.50 and above the group mean were categorized as high SRL and those who scored below 3.50 were classified as low SRL. A CPPT pre-test which consists of ten items used in section A of the immediate post-test was administered to the participants prior to the treatment. The purpose of conducting the pre-test was to obtain baseline data and to measure the initial differences in terms of programming knowledge between the two treatment groups before the treatment. An immediate post-test of CPPT, covering both theory and practical knowledge was conducted immediately after the treatment to gauge the students' programming recall performance. Prior to it, a set of reliability tests were conducted on the two instruments (pre-test and immediate post-test of CPPT) used in order to determine the Cronbach's Alpha reliability coefficients. The inter-rater reliability test was conducted on the CPPT pre-test and immediate post-test because these CPPT tests consist of open-ended questions. The scores from the first examiner and second examiner were then compared to determine the consistency of the rates estimated in the Cronbach's Alpha reliability coefficients. The reliability values for:- (i) the pre-test is 0.915 and (ii) the immediate post-test is 0.954.

Data Collection Procedures

The first year semester one computing students in the two intact classes were involved in the seven-week experimental study. They were randomly assigned to the two treatment groups. The students in the first treatment group (MPP) received the combination of metaphors and pair programming instructional strategy in learning the basic programming concepts. In the control group (PP), the students were exposed to PP as the cooperative learning instructional strategy in solving the programming problems. Topics related to basic C++ concepts such as variable declaration, assignments, three types of control construct and object oriented concepts were covered in the class sessions. To understand the abstract concepts, the students in both groups were taught using programme flowchart and Pseudocode. In the practical session, the students were given weekly tutorial tasks that were assessed on the programming syntax, semantics and pragmatics knowledge of the C++ logic. These tasks required the students to work in pairs to write a working C++ codes based on the given problem scenarios. Thus, they were to apply C++ programming language in converting these logical concepts into working programme codes. On the other hand, they were to derive the logical solution using program flowchart or Pseudocode before converting these logical flows into C++ codes. In each group, the lecturer acted as a facilitator. The explanation on the workable solution and the methods of deriving it were presented by each pair. The purpose of the presentation was to ensure that the students understand the logical flow of the solution generated. The tutorial tasks were designed based on the level of the conceptual and syntactical understanding as highlighted in the McGill and Volet's (1997) programming conceptual framework. Working on the program logic design, the students used either program flowchart, Pseudocode or both. During the practical session, they were required to write part of the programme segment or a complete programme in C++ language based on the logical design (program flowchart and Pseudocode) created.

The students in both the experimental and control groups were paired and each member of the pair was randomly assigned with a role, either as a driver or a navigator. The explanation regarding the roles (driver or navigator) of each member in the pair was given to both the MPP and PP groups. On every programming problem, they were persistently required to cooperate on the same design, algorithm, coding and testing. The role between the driver and the navigator was switched periodically. The experiment was carried out for seven weeks. The immediate post-test was administered to both groups immediately after the treatment. The CPPT instrument was used to measure the students' recall performance of the computer programming knowledge.

Research Findings

In this study, SPSS 17.0 for Windows was used to analyse the scores collected from the two CPPT namely the pre-test and immediate post-test. The ANCOVA statistical technique was applied in order to determine any significant difference between the students with different self-regulated learning level on their programming recall performance. In this study, there were only (i) one independent variable with two methods (MPP and PP), (ii) one dependent variable – recall performance, and (iii) one moderating variable – the level of SRL (high or low). Thus, ANCOVA was used to examine the initial differences between the two groups before the treatment. In order to determine the differences, pre-test score was used as the covariate. This was to ensure that the participants were homogenous in their performance prior to the treatment.

The analysis results are shown in Table 1 and Table 2. Table 1 reveals the ANCOVA findings, while Table 2 shows the descriptive analysis and Table 3 indicates the post-hoc results. The interaction effect between the treatment groups and SRL is shown in Figure 5.

Table 1: ANCOVA Results for the Recall Scores of the Two Treatment Groups

Dependent variable	df	Mean square	F	Sig.
Recall (immediate post-test)	3	1102.28	37.96	0.00*
Group * SRL	1	68.20	2.24	0.14

*significant at 0.05 level

Table 2: Descriptive Statistics for the Recall Performances of the Two Groups with Different SRL Levels

	Groups	SRL	N	Mean	SD
Recall (immediate post-test)	MPP	High	24	74.17	4.08
		Low	18	62.80	4.72
	PP	High	27	69.52	6.94
		Low	15	55.11	6.22

ANCOVA results in Table 1 clearly indicate a statistical significant difference in recall performance between the high SRL and the low SRL students who received different treatment methods ($F = 37.96$; $p = 0.00$). Thus, these findings have rejected both the first and second hypothesis. The post-hoc test was conducted to further investigate the differences (Table 3). However, the graph in Figure 5 reveals no significant interaction effect between high and low SRL students taught in the MPP and PP groups.

Table 3: Summary of Post-Hoc Test for Recall Performance between the High and Low SRL Students in the Two Treatment Groups

Level of SRL	Groups	Mean Difference	p-value	Results
High	MPP vs PP	4.65	0.03	Sig.
Low	MPP vs PP	7.68	0.00	Sig.

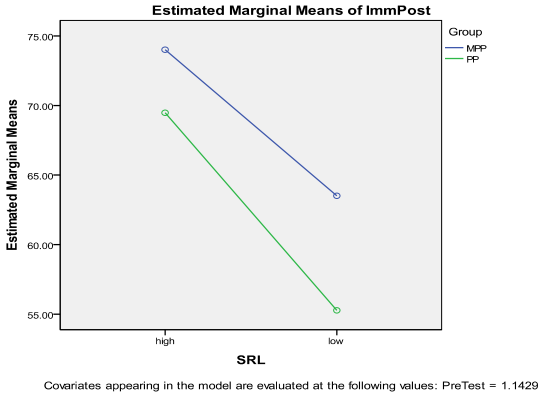

Figure 5: Interaction Effect between the Instructional Methods and SRL

Hypothesis 1: There was no significant difference in recall performance between the high students taught in the MPP and PP groups

The post-hoc test result (Table 3) indicated a significant difference in recall between the MPP and PP groups for the high SRL students, with the former performed significantly better than the latter ($\overline{X}_{highMPP} = 74.17$; $\overline{X}_{highPP} = 69.52$; Mean diff = 4.65; p = 0.03). Thus, the first hypothesis was rejected.

Hypothesis 2: There was no significant difference in recall performance between the low SRL students taught in the MPP and PP groups

The post-hoc result in Table 3 revealed a significant difference in recall performance between the low SRL students in MPP group and those of the PP group (Mean diff = 7.68; p = 0.00), with the MPP group performing significantly better than those of the PP group ($\overline{X}_{LowMPP} = 62.80$; $\overline{X}_{LowPP} = 55.11$). Thus, the second hypothesis was also rejected.

Hypothesis 3: There was no interaction effect between instructional methods and self-regulated learning level

Figure 5 shows that there is no interaction effect between instructional methods and the students' SRL level on programming performance between the MPP and PP groups (F = 2.24; p = 0.14). This would mean that regardless of SRL level, MPP method is much better than PP. Also, high SRL students outperformed the low SRL students in each method Therefore, the third hypothesis was accepted.

DISCUSSIONS

This study aims to investigate the impact of different SRL levels on the students' recall performance on the instructional methods used in learning C++ programming language. These students from the two intact groups were randomly assigned to two different instructional methods (MPP and PP). One group received the MPP treatment and the other was treated with the PP method. The research findings indicated that the difference in the recall performance for high and low SRL students between the two instructional methods were significant. However, no significant interaction effect between instructional methods and SRL was shown. Further analysis revealed that the high SRL students in the MPP group performed significantly better than their peers in the PP group for the programming recall performance. Similarly, the low SRL students taught in the MPP group significantly outperformed those in the PP group. As such, the MPP instructional method significantly influenced on immediate recall for both high SRL and low SRL students.

The metaphors with pair programming instructional method significantly aid both high and low SRL students in visualizing the abstract concepts – either in pictorial or textual forms, thus creating higher mental models for reasoning and engaging in interactive discussion. Therefore, this finding demonstrated that metaphors facilitated and improved learning towards information recall (Flanik, 2008). The use of metaphor supported the formation of memory images of the new programming concepts being introduced and positively influenced on memory recall for both the high and low SRL students taught in the MPP group as compared to the PP group. These high and low SRL students in the MPP group applied metaphorical concepts to connect their current knowledge with the new knowledge that accommodate both sources (target and source) in resolving programming problems, enhancing their understanding and programming comprehension towards recall performance. The metaphorical theory generated the link between the target and source dealing with the transfer of procedural knowledge from one domain to another within the McGill and Volet's (1997) conceptual framework. By progressing from one domain knowledge level to another within the conceptual framework, it gave the students an opportunity to improve their programming performance by enhancing their ability to design, code and test a programme to solve novel problems. The high SRL students are those who set goals for their learning, and independently manage time and effort spend on learning C++ concepts. They are highly motivated and capable of establishing relations between the target and source. With the interaction between current knowledge and novel concepts, it allows the students to build clearer mental images during the mental processes. As they progress through programming tasks, they plan and carry out the learning activities towards the desirable achievement. In line with Wolters, Pintrich and Karabenick's (2003) findings, high SRL students taught in MPP group achieved better understanding and overall recall performance in relation to learning C++ language that those in the PP group; whereby the SRL learning activities cultivate the MPP students to learn the basic C++ concepts in more tacit ways and organize their thinking in an explicit manner. Through classroom and practical learning, metaphors allowed the students to connect their current knowledge and experiences with novel problems; and thus assist the development of a self understandable neural network in their memory. This network of information stored is easily retrieved as ideas amassed from building clearer mental schemas. Likewise,

metaphors assisted both the high and low SRL students taught in the MPP group to view the abstract concepts from across the programming spectrum (problem, design, coding and maintenance) and see visual presentation cues to identify the important target and source, in order to construct a solution based on the given problem scenarios without looking at the individual programming syntax and line. Through this technique, the students in the MPP group were to build on their existing knowledge foundation by mapping current understanding to abstract concepts and then enabled them to recognize the interactions amongst the programming lines. Subsequently, it fostered positive improvement in programming comprehension and recall performance.

For the low SRL students, metaphors assisted those taught in the MPP group to set relationship between the unknown and the known knowledge that linked the two conceptual domains together. This further promoted meaningful learning and enhanced memory recall as they were able to link what they know to the newly introduced concept. As such, the students in the MPP group had better understand and enhanced their programming skills in solving programming as compared to those of the PP group. The rapid assimilation of new ideas by associating new novel concepts with the existing knowledge fostered the development of the mental schemas during the process of leaning programming. Subsequently, it increased the low SRL students' programming comprehension and developed higher logical thinking skill.

The use of pair programming as cooperative learning approach provides the opportunity for the students worked in pairs to discuss, brainstorm ideas and cross check programming codes. For the low SRL students, effective learning takes place when they learn through positive peer pressure in a fun and joyful environment. Since these students in the MPP and PP groups had to work in pairs, they were able to discuss, find solutions for specific problems, form ideas and opinions with their partners (high SRL), and thus helped to cultivate problem solving skills, higher order thinking skills and improved their attitude towards programming (Hawi, 2010). Working in pairs enhances the low SRL students' understanding of the programming concepts expression as these students taught in both the MPP and PP groups are encouraged to interact. The approach of learning that allows them to discuss and self-explain has somehow facilitated their problem solving processes. By making arguments and accepting constructive criticisms from their peers, it does develop higher thinking skills. This type of verbalization approach has resulted in achieving greater level of understanding and did develop clearer "mental model" of the abstract concepts which are crucially important for problem solving (Goel & Kathuria, 2010). In other words, the low SRL students participated in the discussions by explaining each other's approaches to problem solving thereby creating a higher level of conceptual understanding and promoting critical thinking skills that subsequently improved their recall performance (Flanik, 2008; Felder, 1996). Likewise, these students benefited the most from participating in heterogeneous pairs, specifically by offering further explanations to their peers. Similar results were also reported by Meseka, Nafziger and Meseka (2010) as well as by Ballantine and Larres (2007).

This finding revealed no interaction effect between instructional method and the students' SRL level on programming recall performance between the MPP and PP groups. In other words, regardless of SRL level, MPP method is much better than PP. Also, high SRL students significantly outperformed the low SRL students in each method. To enforce effective learning, lecturers should consider the combination of metaphor and pair programming to be adopted in class lectures and during practical session (where the conversation of programming logic into C++ application) as well as to take note of the students' SRL levels in order to have significant influence on their programming performance.

LIMITATIONS AND RECOMMENDATION

This study used a population sample as the number of students registered for that semester was 84. The scope was confined to students of first year computing course at one selected private college in the Northern Region of Malaysia. Therefore, the study cannot be generalised to all Introduction to Programming with C++ students. The students' attitude towards the instructional methods may reflect and influence the overall programming comprehension process. Limited training duration on metaphors and pair programming was also the constraint of this study. It does not permit extensive, detailed and longer training sessions on pedagogy used. The level of problem solving ability and logical reasoning as well as the prior knowledge on programming languages were unknown. Therefore, some of the students had difficulties understanding and comprehending the C++ programming course during the lectures and others found the explanations too simple. Future research could be generalised to local public and private educational institutions in the country. An investigation into the degree to which the characteristics of the participating students influence their attitude towards instruction methods should be considered. For future studies, the duration for the training sessions on metaphors and pair programming should be taken to consideration in order to improve and enhance the learning experience. As critical thinking skill includes problem solving and logical reasoning required in learning programming, it is recommended for the lecturers to incorporate critical thinking skill and also to encourage the students to apply SRL in learning

programming.

CONCLUSION

The findings revealed that metaphor when combined with pair programming has significantly helped students' learning, both for the low and high SRL students. This study has also emphasised the importance of considering SRL components in learning the basic programming concepts through C++ language for classroom academic performance. The importance of adopting metaphor in learning C++ concepts for solving novel problems has been revealed in this finding. Metaphors have performed an essential role in helping the students to create clearer mental images in solving abstract concepts. For problem solving, metaphors develop better conceptual understanding by linking the known to newly acquired abstracts; and pair programming does cultivate peer discussions. Pair programming as cooperative learning strategy has been applied in software engineering industry to increase productivity. When used effectively, pairing activity did further enrich programming knowledge and enhance performances. The combination of metaphor with pair programming is effective in supporting students recall performance. Deeper understanding in applying self-regulation will allow the lecturers to encourage self centered learning activities that will generate positive learning outcomes in terms of program solving skills and programming performance. It encourages the students to identify their strength, weaknesses and to have better understanding of their learning abilities. By applying SRL strategy, it helps the students to determine own learning pace and cycle, which subsequently trigger positive accomplishment in programming recall performance. As such, lecturers should assist students to form conceptual visualisation in their working memory during their teaching in order to reinforce self learning. Self-regulation is the predictor of programming performance which used the self-regulating strategies, for example, the gold setting, planning, time management, self monitoring and evaluation for strengthening the programming knowledge of students and improving their programming performance. As such, the students' programming performance is correlated with the application of instructional methods in course delivery and the different level of SRL of students. When used effectively, these self-regulatory strategies could stimulate students' recall performance. It is suggested the lecturers should encourage their students to apply SRL in programming contexts in order to reinforce self learning. In turn, it promotes the development of knowledge and competency within self through life-long learning process.

REFERENCES

American Heritage Dictionary Editors (2000). *The American heritage dictionary of the English language (4th ed.)*. Boston, Mass: Houghton Mifflin Harcourt.

Ballantine, J. & Larres, P. M. (2007). Cooperative learning: A pedagogy to improve students' generic skills? *Education and Training*, *49*(2), 126-137.

Beck, K. (2000). *Extreme programming explained: Embrace change*. Reading, MA: Addison-Wesley.

Bruce, C. & McMahon, C. (2002). *Contemporary developments in teaching and learning introductory programming: Towards a research proposal*. Queensland University of Technology.

Butler, M. & Morgan, M. (2007). Learning challenges faced by novice programming students studying high level and low feedback concepts. *Proceeding of Australasian Society for Computers in Learning in Tertiary Education (ASCILITE)*, Singapore, 99-107.

Chung, R. G. & Lo C. L. (2006). The study of different cooperative learning and problem-based instructions in promoting students' teamwork competences. *World Transactions on Engineering and Technology Education*, *5*(3).

Felder, R. M. (1996). Active-inductive-cooperative Learning: An instructional model for chemistry? *Journal of Chemical Education*, *73*, 832-836.

Flanik, W. M. (2008). Conceptual metaphor and US missile defence: Preliminary theorizing and analysis. *The Annual Conference of the Canadian Political Science Association*, Vancouver, BC, 1-20.

Goel, S. & Kathuria, V. (2010). A novel approach for collaborative pair programming. *Journal of Information Technology Education*, *9*, 183-196.

Hawi, N. (2010). The exploration of student-centred approaches for the improvement of learning programming in higher education. *US-China Education Review*, *7*(9), 47-57.

Jossberger, H., Brand-Gruwel, S. & Boshuizen, H. (2006). *Self-directed learning in prevocational secondary education: An analysis of difficulties and success factors in workplace simulations*. Open University of the Netherlands.

Kerka, S. (2005). Applying adult learning theory: Self-directed learning and transformation learning in the classroom. *California Adult Education, Research Digest no. 3*, Adult development.

Lakoff, G. & Johnson, M. (2003). *Metaphors we live by*. Chicago: The University of Chicago Press.

Lee, T. H., Shen, P. D. & Tsai, C. W. (2008). Applying web-enabled problem-based learning and self-regulated learning to add value to computing education in Taiwan's vocational schools. *Educational Technology & Society*, *11*(3), 13-25.

Mayer, R. E. (2003). Theories of learning and their application to technology. In H. F. O'Neil & R. S. Perez

(Eds.). *Technology application in education* (pp.127-157). Mahwah, New Jersey: Lawrence Erlbaum Associates.

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.

Merwe, D. V. D. (2008). *Interpretation and visualization of C/C++ data structures*. Unpublished Degree thesis, Rhodes University. Retrieved October 10, 2010, from http://research.ict.ru.ac.za/g05v0090/CSHnsThesis.pdf

Meseka, C. A., Nafziger, R. & Meseka, J. K. (2010). Student attitudes, satisfaction, and learning in a collaborative testing environment. *The Journal of Chiropractic Education*, *24*(1), 19-29.

Miliszewska, I. & Tan, G. (2007). Befriending computer programming: A proposed approach to teaching introductory programming. *Issues in Informing Science and Information Technology*, *4*, 277-289.

Parker, P. M, (2009). *Webster's online directory: With multilingual thesaurus transaction*. Insead. Retrieved December 21, 2009, from http://www.websters-online-dictionary.org

Pintrich, P. R. (2000). The role of goal orientation in self-regulated learning. In M, Boekaerts, P. R. Pintrich & M. Zeidner (ed.), *Handbook of self-regulation (pp. 451-502)*. San Diego, CA: Academic Press.

Pintrich, R. R. & DeGroot, E. V. (1990). Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology*, *82*, 33-40.

Reyero, M. & Touron, J. (2003). *The development of talent: Acceleration as an educational strategy*. Spain: Netbiblo.

Tie, H. H. (2011). *The effects of metaphors and pair programming on recall and retention amongst students with different learning styles and self-regulated learning levels*. Unpublished PhD thesis, Universiti Sains Malaysia.

Wolters, C., Pintrich, P. R. & Karabenick, S. A. (2003). Assessing academic self-regulated learning. *Paper presented at the Indicators of Positive Development Conference*, Washington, DC, March 12-13, 1-49.

Zimmerman, B. (2008). Investigating self-regulation and motivation: Historical back-ground, methodological development, and future prospects. *American Educational Research Journal*, *45*(1), 166-183.

Zimmerman, B. J. (2000). Attaining self-regulation: A social cognitive perspective. In M. Boekaerts, P. R. Pintrich & Zeidner (ed.), *Handbook of self-regulation (pp. 13-39)*. San Diego: Academic Press.

