INSTRUCTIONAL STRATEGY IN THE TEACHING OF COMPUTER PROGRAMMING: A NEED ASSESSMENT ANALYSES

Mohd Nasir ISMAIL, [1]Nor Azilah NGAH, [2]Irfan Naufal UMAR

Faculty of Information Management
Universiti Teknologi Mara Cawangan Kelantan
18500 Machang, Kelantan
nasir733@kelantan.uitm.edu.my

[1, 2] Center for Instructional Technology and Multimedia
Universiti Sains Malaysia 11800 Minden, Penang
azilah@usm.my, irfan@usm.my

ABSTRACT
The process of Instructional Design deals with the production of an effective, efficient and appealing instructional material under different condition, method and outcome. Computer programming is part and parcel of computer education. Research done in western countries has shown that programming requires problem solving and analytical thinking skill; unfortunately these skills are found to be deficient among many students pursuing computer programming courses. A needs assessment was done to identify whether such a problem exists amongst Malaysian students pursuing computer programming courses in a Malaysian university. Among others, the aim of the needs assessment is to identify the instructional problems pertaining to the current strategies used for the teaching of programming. This paper reports and discusses the findings collected from the interviews with five computer science lecturers from the faculty of computer science in a local university. The result shows that there are deficiencies in knowledge, understanding and application of computer programming among computer science students. Recommendations are given for further investigation into a more effective strategy as an alternative in the teaching of computer programming courses.
Keywords: Computer Programming, Computer Education, Needs Assessment, Instructional Strategies, Instructional Design

INTRODUCTION

Computer programming is part and parcel of the computer science education. It is an essential skill that must be mastered by anyone interested in studying computer science. Normally, in teaching computer programming, students will first be introduced to the concept of programming and data structure where they are taught on how to analyze problems, use specific techniques to represent the problem solution and validate the solution. Next the learners are required to convert the problem solution into a program using a specific programming language. They are then required to test their program to verify for syntactical or logical errors to ensure that the output is correct according to the problem requirement. Maintenance is the last process in implementation phase and it is based on user requirement needs. Maintenance is required when there are changes in user requirements or important components. The whole process of computer programming is shown in Figure 1.

Experience in teaching university level computer programming has proven to be a challenge to the first author. Many students found programming to be difficult and disheartening. Since programming is the basic skill required of computer programmers, the negative impact of these basic introductory courses may have harmful consequences in the learners' attitude towards the field.

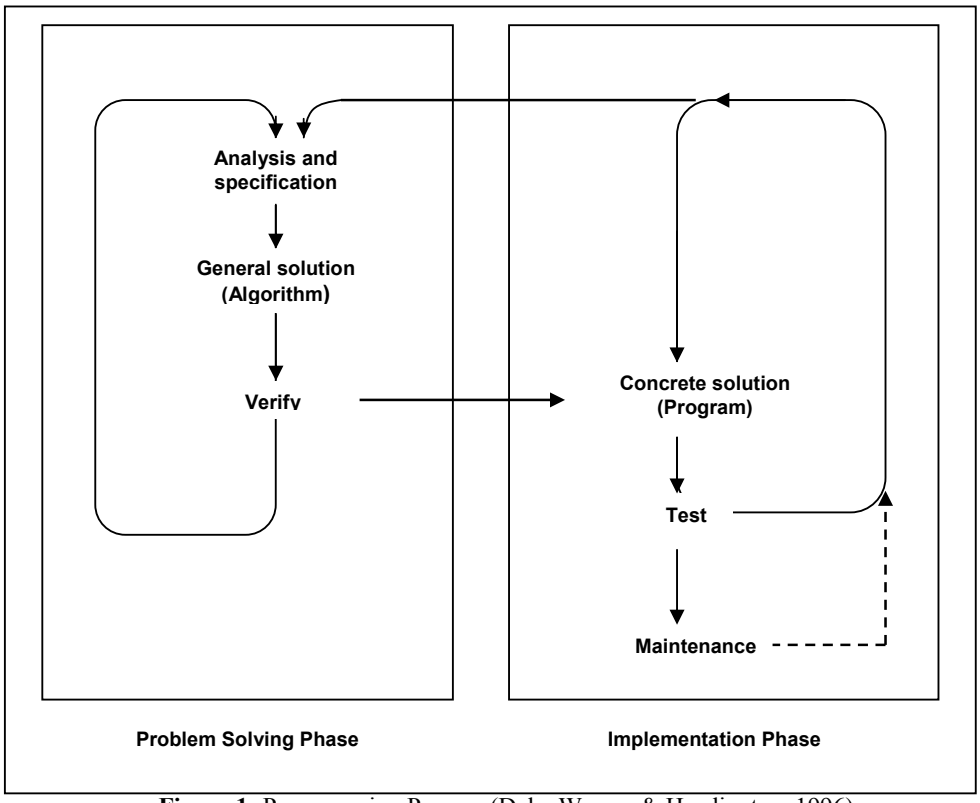

Figure 1: Programming Process (Dale, Weems & Headington, 1996)

PROBLEM STATEMENT

Learners' difficulty with computer programming is not unique to the Malaysian audience. Research done in western countries has shown problems with regard to computer programming. The skills that have been identified with the ability to do programming are problem solving and analytical skills (Riley, 1981; Henderson, 1986; Maheshwari, 1997b; Bonar & Soloway, 1989; Linn & Clancy, 1992). However, according to Riley (1981), many students entering college have problem-solving skills that are "woefully inadequate". Henderson (1986) notes that problem solving and analytical thinking skills are students' major weaknesses in a computer science course and that a major theme of a computer science course should be emphasized on these skills. Programming is said to be a study of clear thinking and problem solving in providing the students the practice of building representations and working in a methodical manner (Maheshwari, 1997b). Maheshwari also argues that programming fosters problem solving through a top-down approach, whereby large problems are separated into manageable components to be solved individually and then assembled into the correct solution to the problem. Programming encourages learners to evaluate their solutions and thinking process; this cognitive process allows them to transfer newly acquired problem solving skills to novel problem situations. Whatever approach to problem solving is adopted, it is recognized that it is an essential part and the first step taken in the development of software.

In addition to problem solving and analytical skills, difficulty in programming is also attributed to the prior knowledge and practices; errors also occur in trying to transfer a step-by-step problem-solving solution directly from a natural language into a program (Bonar & Soloway, 1989). The differences between the natural language and a programming language can easily cause problems. For example, some novices have understood that the condition in a "while" loop needs to apply continuously rather than tested once per iteration. Linn & Clancy (1992) found that "for programmers to develop competency, they need to have good problem solving skills and a thoroughly organized knowledge of a programming language". In problem solving phase, a solution or design is generated to solve the problem and in the implementation phase the proposed solution is translated into a programming language. According to Rist (1996), the main source of difficulty does not seem to be only on the syntax or understanding of concepts, but rather on the program planning. A student can learn to explain and understand a programming concept, e.g., what does a pointer mean, but still fails to use it appropriately in a program. Winslow (in Soloway & Spohrer, 1989) noticed that students may know the syntax and semantics of

individual statements, but they do not know how to combine these features into valid programs. Even when they know how to solve the problem manually, they have trouble translating it into an equivalent computer program.

Most of the introductory text books on computer programming emphasize on the study of a programming language; the pre-programming topics such as introduction to algorithmic (pre-coding), and the running of programs on a computer are eliminated. According to Gal-Ezer (1996), even if a lecturer has introduction to algorithmic in mind, the emphasis in practice is always on the technicalities of a programming language, coding and running programs on a computer. Linn and Clancy (1992) claimed that most introductory programming language textbooks reinforce the emphasis on syntax and on the learning of individual examples rather than encouraging students to recognize and reuse more complex patterns. McGill and Volet (1997) found that most of the introductory computer programming courses and text books only emphasize on lower level knowledge or known as declarative knowledge and procedural knowledge that emphasize on "know that" and "know how" that are related to programming concepts and syntax. As a result, students fail to understand and are not able to explain semantics actions in a program. The emphasis on low level knowledge will cause students not to understand and master the programming syntax and constructs. Thus, learners are not able to apply correct rules of syntax during programming and are not able to use semantic knowledge of programming in writing program to solve novel problems.

Most programming courses are taught using the traditional approaches including a blend of lectures, reading and practical sessions (Gray, Boyle & Smith, 1998). The environments for these types of approaches will only produce students who are passive information receivers, allow minimal interaction between teacher and students especially when a large group of students is involved. Gage and Berliner (1992) also argued that this type of lecturing is not appropriate if specific goals and objectives need to be addressed, need long period of information retention, the learning materials are complex and abstracts, students participation in class are essential to achieve learning objectives and higher level of cognitive objectives (analysis, synthesis and evaluation) are the purpose of the instruction.

Dalton and Goodrum (1991) have suggested that computer programming and problem solving strategy instruction, when used together may provide an effective means of teaching transferable problem solving skills. Maheshwari (1997a) also suggested that programming lessons should employ systematically designed direct instruction activities, rich in feedback and practice opportunities. Programming activities should be designed to encourage the application of problem solving strategies such as planning, simplification and modeling. She also stated that lessons should quickly develop a rudimentary mastery of language syntax and move quickly to produce application and problem solving. In other words, teaching programming should be interesting, motivating and stimulating for both students and lecturers.

The first author's experience as a lecturer in computer science field has shown that students need to acquire reasoning, analytical thinking and problem solving skills for analyzing problem before they learn how to use and apply problem representation tools and computer programming languages. The students need to understand how to interpret the given problem before they can represent the correct solution and effectively use specific tools or techniques. The later skills can be acquired by doing a lot of practices in problem solving that involved planning, logical thinking and reasoning strategies. However, mastery in the reasoning and problem solving skills does not necessary mean that students are able to write good computer program as writing programming languages requires the mastering of the syntax and functions of the specific programming languages. Mastering of these elements require the students to be actively engaged in practical exercises in writing program by using correct syntax and constructs.

Students usually react passively during lecturing and tutorial session and this makes assessment of student's mental understanding difficult. At the same time, they believe that computer programming skill is complex and difficult to be acquired and this could hinder them from asking questions for clarification. Usually, students who are able to acquire the programming skill are those who are highly motivated and interested in exploring the programming problems. They are usually actively involved in class and always seek help and discuss any problems relating to computer programming with their lecturers and colleagues. Table 1 shows the problems identified in the literature concerning problems in computer programming.

Table 1: Problems in computer programming as identified in the literature

Problem Solving Phase		Implementation Phase
Analysis	**General Solution**	**Detail Solution**
• Lack of problem-solving skills • Lack of analytical thinking skills • Lack of logical and reasoning skills • Lack of programming planning • Lack of programming conceptual understanding • Lack of algorithmic skills	• Inefficient tools used in representing problem solution • Do not understand and unable to explain semantics actions in a program • Ineffective design and testing problem solution	• Do not understand and master the programming syntax and functions • Unable to apply correct rules of syntax when programming • Unable to use semantic knowledge of programming to write program • Ineffective code and testing program to solve novel problem

OBJECTIVE

The main aim of this research is to identify the problems in computer programming education in Malaysia. A need assessment was conducted to identify problems relating to teaching and learning programming and finding possible solutions to this problem. The paper will present the result of this need assessment.

METHODOLOGY

Participants

The needs assessment was done by collecting data from interviews with five expert lecturers in computer science field at a local university. An interview protocol to elicit information on the problem under discussion was created and used as a guideline during the interview sessions. The participation was voluntary in nature and each interview session was around an hour to two hours.

Five university lecturers participated in the study. The selection of the participants is based on year of experience in teaching computer science programming courses. Two of them are doctorate and the others are master degree holders. Four of the participants have been teaching for more than ten years; meanwhile, the other one has seven years of teaching programming with vast experiences in software engineering, managing a software development company involved in developing commercial computer application systems. The lecturers are experienced in teaching various types of programming languages and paradigms such as C language for structured programming, C++ for object-oriented programming and Prolog and LISP for logic and artificial intelligence programming language at both the undergraduate and graduate levels. Two of the participants are supervising doctoral students at the university. They are also actively involved in research projects and consultations regarding software engineering, artificial intelligence, parallel processing et cetera.

Interview Protocol

An interview protocol was developed to elicit information concerning the lecturers' perception on the importance of students' understanding of programming concepts, problems and causes of problems in learning programming. In addition to identifying the problems faced by students in computer programming courses, the expert participants were also asked to talk about the solutions, methods and strategies they used as suggestions to their students and used by them in overcoming some of the problems identified.

FINDINGS AND DISCUSSION

In this section, the findings from the needs assessment are discussed. Basically, the four main problems were identified by the expert participants. A summary of the problems is shown in Table 2 and the following discussion will be based on these four main problems, solutions to some of the problems identified by the experts and recommendation by authors on some research possibilities as the solutions for some of these problems.

Problem Type I: Lack of Skills in Analyzing Problems

All the five experts interviewed agree that students' understanding of problem solving concepts in a programming course is essential for them to learn programming languages. They said that the lack of understanding of the programming concepts at most basic problem solving level will cause difficulty in the students' further understanding of programming syntax and functions. The experts believe that most students take the skills in problem solving for granted and fail to identify their programming weaknesses at this level. However, the experts disagree on the reasons behind the lack of these skills in this area.

Table 2: Problems identified in the needs assessment process

Problem Type	
I.	Lack of skills in analyzing problems
II.	Ineffective use of problem representation techniques for problem solving
III.	Ineffective use of teaching strategies for problem solving and coding
IV.	Do not understand and master the programming syntax and constructs

One expert believes that the students should be introduced to a course in discrete mathematics and logic before taking any course in programming. In other words, the students do not have the prerequisite skills to take programming courses. Three of the experts said that the students were not actually taught and exposed to proper algorithm solution as the goals for most programming courses are for the students to be able to write programs. Understanding the programming concepts and semantics behind the program were assumed to be acquired by doing the programming exercises.

Suggestions by the experts to solve the problems at this phase of programming include the need for the students to acquire problem solving, planning, discrete mathematics, logic, and creative thinking skills before they learn programming concepts.

Problem Type II: Ineffective Use of Problem Representation Techniques for Problem Solving

According to the expert participants, at the basic level of programming (problem solving phase), two-way discussion approach is used to discuss the definition statement of programming problem. After defining the problem statement, problem solution are usually designed using algorithm representation techniques. Techniques such as pseudo code and flow chart are used to present the algorithm during problem solving phase. Both techniques are the accepted standard or conventional techniques and are used to explain the concept of programming in most Malaysian universities. The same techniques are also being used in the computer programming books written by the authors from western countries. Both techniques are based on structured problem solving method whereby a problem is presented in a form of procedural statements similar to the actual programming code (pseudo code) and presented in a form of control flow or data flow process (flow chart). At this phase, the problem appears to be similar to the type of programming codes that are being taught to the students.

All expert participants agreed that the conventional techniques used to represent the algorithm have created some problems for the students, especially for those doing object-oriented programming. According to them, these conventional techniques are more suitable for structured programming approach and can cause the students to be confused and unable to translate the algorithm into the correct programming coding. They also agreed that the concept of programming that is based on object oriented approach should be introduced to the students in semester two, that is after they have already grasp the foundation on structured approach. Also, according to them, the object oriented approach is best used to explain a problem in a form of program entity. Furthermore, at the basic level, most of the experts interviewed agreed that concept programming that uses structured approach is much easier to understand by the students since this is the approach human use in thinking.

Some of the solutions suggested by the experts include the use of different problem representation tools for different types of programming. This is to say that structured programming approach should use a different problem representation tools than object oriented approach. The instruction should also be supported by using visualization approach that would enable the students to have a mental representation of the problem. Lastly, the time spent for the teaching of concepts of programming should also be made longer to about 3 or 4 weeks. Currently, the time spent for teaching the concepts of programming is only about 2 weeks.

Problem Type III: Ineffective Use of Teaching Strategies for Problem Solving and Coding

Three of the expert participants claimed that the difficulty in understanding the concept of programming and coding is because of the ineffective teaching strategies used during problem solving and coding. These experiences will undoubtedly influence the students' perceptions on programming courses as difficult and complex. One expert participant argued that factors such as lecturer using ineffective teaching strategies and taking the matter into granted contribute to the difficulty in understanding and confused the students when they try to apply the concept into programming code. According to this expert participant, the effective teaching strategies should start with teaching structured or procedural type of programming language; object-oriented type of programming language is not a good starting point to introduce the students to the basic concept of programming. Two other expert participants believe that the main cause for the above problems is the inactive involvement of students during programming practical session.

All the expert participants also agreed that the concept of programming should be taught to the students in a form that support their spatial and visualization abilities as these aspects will help them to understand and visualize the process of control and data flow in a program in a more general context. All of them agreed that techniques, approaches and strategies used in teaching programming should be applicable to the content of programming with different paradigms in order to help students strengthen their basic problem solving skills and be able to plan and organize the solution by using an effective cognitive strategy. The cognitive strategy will hopefully help them to acquire the problem solving skills that together with knowledge on the syntax of a programming language can help them to solve novel problems.

Some of the problems suggested by the experts include doing enough practical exercises relating to real world examples as these would allow them to apply the concept of programming correctly to solve novel problem. Practical sessions or tutorial should also be enriched with activities, feedback and practice opportunities.

Problem Type IV: Do Not Understand and Master the Programming Syntax and Constructs

According to the experts, students need to have both the understanding of the concept of programming and the knowledge of syntax and constructs of a specific programming language in order for them to be able to write a good program. They added that lecturers normally give lectures on the concepts and principles of programming along with simple examples of problems and provide students with practical exercises to build program concepts and translate them into programs. Practical exercises are done in the computer laboratory during tutorial sessions. For the weak students, they are urged to make appointment for consultation or create small group remedial session to help them overcome these problems. The experts also added that practical exercises are important and students should be active participants during these tutorial sessions and should spend time understanding the syntax, construct, and concept of the programming languages.

In order to overcome these problems, the experts have also suggested the collaborative and cooperative group work amongst the students. Team work allows for the use of scaffolding and coaching on how to programming effectively thus allowing them to explain and understand the programming concept, know the syntax and semantics of programming statements and know how to combine these features into valid computer programs.

DISCUSSION AND CONCLUSION

Analyses of the data from the needs assessment revealed some similarities between problems identified by the expert participants and the first author's experience in teaching similar courses. There are gaps or deficiencies in students' knowledge in computer programming course in each phase of the programming processes. Four main problems were identified, including (i) the lack of skills in analyzing problems, (ii) ineffective use of problem representation techniques for problem solving, (iii) ineffective use of teaching strategies for problem solving and coding, and (iv) the difficulty in mastering programming syntaxes and functions.

According to McGill and Volet (1997), most introductory computer programming courses and text books emphasize only the lower level knowledge, also known as declarative and procedural knowledge. Declarative and procedural knowledge are types of knowledge that emphasize the knowledge of "what" and "how" respectively. As such, these are knowledge that are related to the what and how of programming concepts and syntax. Rist (1996) believes that the acquisition of only low level knowledge made it difficult for students to apply a complete form of programming even though they are able to explain and understand the programming concept. This will cause the development of inert knowledge to the students during the learning process. This is the same observation made by Winslow (in Soloway & Spohrer, 1989) where he noticed that students may know the syntax and semantics of individual statements, but they do not know how to combine these features into valid programs.

Computer programming requires higher level knowledge or knowledge at the strategic or conditional level. This is the knowledge of "when and why" which requires meta-cognitive skills which are apparently are lacking among the students. Lack of meta-cognitive skills has been reported in several studies on computer programming courses (Linn, 1985; Linn & Clancy, 1992; McGill & Volet, 1997; Oliver, 1993; Volet, 1991). If one were to look at the different phases of the programming processes as shown in Table 1, even at the initial and first phase of problem solving, analysis of the problem requires the student to be able to identify, analyze, plan and create possible ways to put the problem into whatever programming language at hand, a task that requires the highest cognitive dimension identified in the Revised Bloom Taxonomy (Anderson & Krahwohl, 2001). The experts' opinion from this needs assessment concur with the literature on computer programming education in that the critical part of the programming process starts at the analysis of the problem solving and consequently will have an effect on the next phase of the programming sequence.

Is there a teaching or learning strategy that can be used to help lessen the burden at this stage? Is there a need for a specific kind of technique to represent the individual's knowledge and understanding regarding computer programming problem? Are pseudo codes and flowcharts adequate in helping the students to see the problem to be programmed? What are some of the visual representations other than the flowchart that can be used at this stage? These are some of the questions that need to be answered and further research need to be done to find the solution. Otherwise our computer programmers in the future will not have the skills necessary to create new applications, merely users of programs created by others. In the era of digital technology and knowledge workers, these are inadequate skills that need to be addressed in the field of Instructional Technology.

REFERENCES

Anderson, L. W., & Krathwohl, D. R. (Eds.). (2001). *A taxonomy for learning, teaching and assessing:* A revision of Bloom's Taxonomy of educational objectives: Complete edition, New York: Longman.

Bonar, J. & Soloway, E. (1989). Pre-programming knowledge: A major source of misconceptions in novice programmers. In Soloway & Spohrer (1989), *Studying the Novice Programmer* (pp. 325-354), Mahwah, NJ: Erlbaum.

Dale, N. et.al. (1996). *Programming and Problem Solving with C++*. Boston: Jones and Bartlett Publishers.

Dalton, D. W., & Goodrum, D. A. (1991). The effects of computer programming on problem-solving skills and attitudes. *Journal of Educational Computing Research, 7*(4), 483-506.

Gage, N. & Berliner, D. C. (1992). *Educational Psychology*. Boston: Houghton Miffin.

Gal-Ezer, J. (1996). A pre-programming introduction to algorithmics. *Journal of Mathematics and Computer Education*. *30*(1), 61-69.

Gray, J. et.al. (1998). Proceedings from ItiCSE '98: *Integrating Technology into Computer Science Education*, pp. 94-97, New York: ACM Press.

Henderson, P. B. (1986). Proceedings of the 17th SIGCSE '86: *Technical symposium on Computer Science Education*, pp. 257-263, New York: ACM Press.

Linn, M. C. (1985). The cognitive consequences of programming instruction in classrooms. *Educational Researchers, 14*(5), 14-16 & 25-29.

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

Maheshwari, P. (1997a). Improving the learning environment in first-year programming: Integrating lectures, tutorials, and laboratories. *Journal of Computers in Mathematics and Science Teaching*, *16*(1), 111-131.

Maheshwari, P. (1997b). Proceedings from ACM International Conference Proceeding Series '97: *Proceedings of the second Australasian Conference on Computer Science Education,* pp. 32-39, New York: ACM Press.

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

Oliver, R. (1993). The contextual model: An alternative model for teaching introductory computer programming. *Journal of Computers in Mathematics and Science Education, 12*(2), 147-167.

Riley, D. (1981). Proceedings from Technical Symposium on Computer Science Education '81: *Proceedings of the twelfth SIGCSE Technical Symposium on Computer Science Education*, pp. 244-251, New York: ACM Press.

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

Soloway, E. & Spohrer, J. (1989). *Studying the Novice Programmer*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Volet, S. E. (1991). Modeling and coaching of relevant metacognitive strategies for enhancing university students' learning. *Learning and Instruction, 1,* 319-336.

