Teaching Programming and Problem Solving to CS2 Students using Think-Alouds

Naveed Arshad
Department of Computer Science
LUMS School of Science and Engineering
Sector U, DHA
Lahore, Pakistan
naveedarshad@lums.edu.pk

ABSTRACT

Many studies have shown that students often face difficulty in applying programming concepts to design a program that solves a given task. To impart better problem solving skills a number of pedagogical approaches have been presented in the literature. However, most of these approaches provide a general strategy of problem solving. But in reality problem solving is a skill that is developed with experience over a period of time. In this paper, we present a pedagogical approach to teach problem solving using think-alouds. In a think-aloud problem solving approach students learn the skill of problem solving by closely observing an 'experienced programmer. We used this approach in a CS2 class and our evaluation results show that think-aloud problem solving is an extremely effective pedagogical technique, particularly for female students.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education - Computer Science Education, Curriculum

General Terms

Experimentation, Design, Measurement

Keywords

Think-aloud, Programming, CS2, Problem Solving

1. INTRODUCTION

At our university we have a CS2 Problem and Problem Solving course in the undergraduate curriculum. This is a required course for all undergraduate computer science and computer engineering students. The pre-requisite for this course is a CS1 course. The CS1 course although has some programming content but it is a more focused course on introducing students to computing as a field of work and study. Therefore, the students get their first hands-on introduction to programming in the CS2 course.

The students who took CS2 course in previous years reported a few problems with the way the course was designed. One of the reported problem was the inability of students to comprehend and solve a given problem pragmatically. What the students said was that they neither have any problems in understanding the concepts of programming language nor in writing its syntax but when a new problem is given to them they have immense difficulty in transforming the given problem into code. In other words the students know the programming language but do not know how to design a problem to solve a given task. This feedback is also validated by results of other studies that evaluated the study of difficulties faced by novice programmers [8, 13].

Therefore, we redesigned the CS2 course and introduced some new pedagogical techniques that were specifically developed to enhance students' problem solving skills. One of the new technique that we introduced is a think-aloud problem solving approach. In this paper we outline this approach alongwith other new or improved pedagogical instruments. Towards the end we provide a comparative evaluation of the effectiveness of all these instruments.

2. COURSE DESIGN

The CS2 course is a quarter long four unit course with two weekly lectures and a weekly lab. This is supplemented by four weekly recitation sessions, two after each lecture and students have the option to attend any one of them. We used a principled OO first approach proposed by Gries [5] to teach programming and problem solving in OO paradigm. The language that we chose to teach in this course was Java. One of the reasons to select Java as a teaching language was the amount of online material available to teach Java. Moreover, we employed many graphical teaching tools to introduce the concepts like recursion, gui programming and so on and Java provides a rich set of third-party libraries to teach these concepts [6]. Although we chose Java as a language of choice in this course the approach that we present in this paper could be applied in any programming course using any programming language.

First year programming courses are difficult to teach in a sense that virtually every student has a different level of programming experience. Some have done programming at high school level and they know the ins and outs of one or more programming language(s). On the other hand there are some students who only have a minimal programming

SIGCSE'09, March 3–7, 2009, Chattanooga, Tennessee, USA.

background. To accommodate as much students as possible we started with an assumption of a zero programming background for all students. Therefore, at the start of the quarter we covered basic programming concepts like variables, decision structures, loops, recursion, classes and objects etc. After almost the middle of the quarter we started covering the more hardcore object-oriented concepts such as polymorphism which included inheritance, interfaces and other concepts such as inner classes and gui programming.

We developed various new instruments to teach students programming and problem solving. The motivation behind developing these instruments was based on the learning theories such as Kolb-based Learning Styles Inventory (KLSI) and Felder's Learning Model (FLM) [7]. Both of these learning theories state that learners learn in different ways and in order to make the course more effective learners need a way that can motivate and keep their interest in the course. Therefore, we developed instruments that cover the various learning styles in KLSI and FLM.

2.1 Programming and Think-aloud Problem Solving

Think-aloud is a protocol widely used in many fields such as usability testing. This protocol was originally designed by Ericcson and Simon [3]. The goal of this protocol is to understand the thought process of a subject in using a product or device. In a think-aloud protocol a subject is given a task to perform and the subject verbally explains the method he or she is employing to complete the task. Therefore, whatever is going on in the mind, the subject is required to speak it out aloud in front of some observers. Usually a subject is observed by one or two observers whose basic goal is to remind the subject to keep talking if the subject stops speaking.

Teaching using think-alouds have been used in various other fields such as Physics [12]. However, we have found very little usage of this technique in teaching programming and problem solving [11]. Also, in literature various problem solving approaches are proposed in introductory programming courses [2, 10]. However, it is also well understood that problem solving is a skill that one learns over time and no single technique or book is suitable to teach problem solving [4, 1].

To this end, we employed a think-aloud problem solving approach that is a kind of imitation-based learning. This learning is based on the suggestions outlined by Caspersen on teaching programming. The gist of these state that teaching programming must be done through revealing the programming process and skills [1]. Therefore, in this technique students learn problem solving by observation. Since problem solving skills of expert programmers are developed over a period of time, our goal was to show the students the underlying thought process that solves a given problem programmatically. Specifically, what kind of approach is employed by the subject in understanding, comprehending, analyzing, synthesizing and coding a problem in a given programming language. Using think-alouds we were able to give an opportunity to students to observe the problem solving process as practiced by an experienced programmer.

To materialize think-alouds for the CS2 course we selected some of the best graduate and senior students as Teaching Assistants (TAs). These students have shown excellent problem solving and programming skills previously. Further, some of these students have had some years of programming experience in industry too. These TAs are then trained in the basic process of the think-aloud protocol at the start of the quarter. The think-aloud protocol that we employed in this course is different in two respects from the original think-aloud protocol. First, the subject in this protocol is a TA in a recitation session being observed by a group of students. Second, in the original think-aloud protocol observers are discouraged to ask any question apart from reminding the subject to keep speaking because it may disrupt the thought process of the subject. However, we encouraged the students to ask questions if they do not understand any particular step that the subject is performing.

Each week in the course we cover some topics in the lectures. After the lecture the TAs perform the think-aloud problem solving during the recitation sessions. In the recitation sessions we asked the TAs to take a sizeable problem that could be solved by the concepts studied in the course so far with particular emphasis on the topics being discussed during the week. During the recitation session the subject's task is to start from the description of the problem and solve it in front of the class while thinking aloud each step that he or she is performing upto the final step of writing the code. This gives the students an opportunity to look at the problem solving technique and see how the subject is breaking down the problem into its constituents, synthesizing the constituents and finally writing the code for the problem.

Not only think-aloud strategy proved to be a great teaching tool in problem solving but it provided many other benefits also. The first of them was the ability of students to write better code. We had observed previously that first year students do not pay attention to coding style and coding conventions. However, while looking at a disciplined programmer they were able to learn the art of writing good code. This was demonstrated in the programming problems given to students later in the quarter.

The second benefit was the ability to design modular programs. Again it is a tendency of many of the first-year students to write all their code in one class and sometimes in just one method of the class. Therefore, while learning problem solving they also get a feel of how pieces of a problem could be broken down into modules or classes and how these modules or classes communicate with each other.

2.2 Labs and Assignments

As it will be discussed later that think-aloud approach is tightly coupled with other teaching tools that we used in this course. Therefore, it is reasonable to provide a description of them.

During the course, to accommodate students with different levels of programming ability, every week we had three levels of programming problems. All the students were required to do these problems but these programming problems differ in terms of their time flexibility. The three types of programming problems were given including a basic labs, an advanced labs and and an assignment or a project deliverable.

2.2.1 Basic Labs

The basic lab session is a two hour weekly session. The basic labs were designed using the lab design approach presented by Robbins and colleagues [9]. The goal of the basic lab session was to give smaller but meaningful programming

problems to students that solidifies the foundational concepts of a particular topic. For example, during the week when we taught recursion in class the basic lab was to program Fibonacci numbers. The students with good learning skills in programming were able to do such problems in an hour or less. Nevertheless, the students who had difficulty in learning programming concepts were also able to complete the lab in the allotted time. In most of these labs we observed that upto 90 percent of the students were able to finish the lab in the allotted time.

2.2.2 Advanced Lab

To build on the foundations set during the basic lab, we used to give another more complicated set of programming problems to students. These were a relatively advanced set of programming problems that required more thinking and sometimes more code writing. However, to make it easier for students who are not fast programmers we allowed the students to complete these programming exercises in an extended time period which is usually forty eight hours following the basic lab. An example of an advanced lab is finding a palindrome or drawing a spiral with some gui programming library such as the one developed by Guzdial and Ericson [6].

2.2.3 Assignments/Project

To solidify the programming and problem solving skills we asked the students to apply the concepts of programming to a new problem set and, if possible, come up with an innovative way of solving the problem. For the first five weeks the assignments were based on the topics of the basic and advanced lab. For example, during the week we discussed recursion we asked the students to build using recursion a graphical binary tree or a Koch's snowflake.

Later in the quarter we asked the students to do a five week long comprehensive project. This project had five deliverables for the five weeks replacing the assignments for those weeks. The project was to implement a photo search engine. The five phases of the project involved a cumulative effort which means that each phase builds on the work of the previous phase. In the first phase the students were asked to learn manipulating the images such as flipping, rotating, etc. In the second phase they were asked to compare two images and to tell if they are similar or not. In the third phase they had to scan a given directory and index the images. In the fourth phase they had to search a given image in the indexed list of images. And finally they had to develop a gui layer on top of the image search engine. At the end of the quarter students were required to present their project in person and explain its salient design aspects.

2.3 Lectures

We used the Kolb Learning Cycle (KLC) to deliver the lectures [7]. In a typical KLC the lecture starts from the 'why' of the topic. In 'why' motivational examples of topic and its importance is discussed. This is followed by the 'what' of the topic where the details of the topic are discussed at a theoretical level. Next is the 'how' of the topic which is where the students are shown the application of the topic to solve a given task. Due to time limitations this is not always possible to show this application in class. Therefore, the recitation session with think-aloud problem solving was used to complement the lecture for the 'why' of the topic. The fourth is the 'what if' of the topic where the students act as devil's advocate of the topic. Since the lectures were already quite interactive students used to ask the 'what if' kind of questions throughout the session but in the end a 5-10 minutes time was reserved for any pressing questions.

2.4 Miscellaneous Instruments

To provide ample opportunities of learning we also used some other learning tools. For example, to foster offline discussion we created a discussion board of the class on Google Groups. The reason for using Google Groups instead of using some other discussion board is because students use Google frequently as a tool and using another tool from the same toolset family is more conducive than using an entirely new tool.

Moreover, we developed self-assessment quizzes for students. After each class a self-assessment quiz was posted on the website for the students to evaluate their own understanding. Students were encouraged to solve the quiz after going through the related readings and before coming to the next class.

Apart from the aforementioned tools we also had a set of readings for each class Moreover, we had instructor and TA office hours for the students. Some of the TA office hours were also scheduled late at night or on weekends to give students maximum opportunity to learn.

3. EVALUATION

This redesigned CS2 course was taught to a class of 120 students with 90 male and 30 female students. At the end of the class we asked the students to fill an anonymous online survey. Out of a class of 120 we received 107 responses, 78 male students and 29 female students filled out the survey form.

3.1 Programming and Problem Solving Ability

The first question we asked was about the programming and problem solving ability. Specifically, we asked the question in the survey: "After taking this course, I feel comfortable in programming a given problem?". Figure 1 gives an overview of the students' response to the aforementioned question. Out of the 107 responses, 61% agree to the statement, 19% percent disagree with the statement and 20% remained neutral.

Upon analyzing this response on the basis of gender in figure 3 it is clear that female agree to the statement more than male students. This is a very encouraging response for us because like many other places we have been dealing with dwindling female enrollments in our computer science and computer engineering programs and one of the reason for this enrollment decline is the frightening factor of learning programming.

3.2 Effectiveness of Pedagogical Instruments

After asking the students about their programming and problem solving ability we then asked the students to rate the various instruments used during the course to teach them programming and problem solving. They were presented with a list of nine instruments i.e. Basic Lab, Advanced Lab, Assignment/Project, Google Groups, Lectures, Instructor and TA Office Hours, Readings, Recitations and Self-assessment Quizzes. The students were then asked to

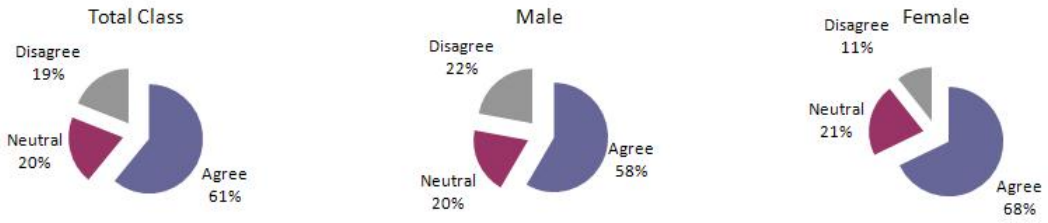

Figure 1: Response of Programming and Problem Solving Ability Question

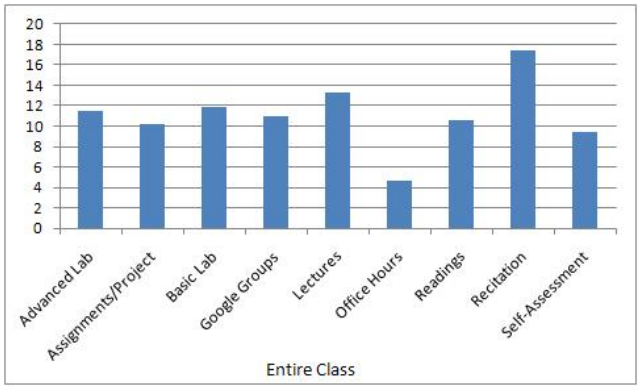

Figure 2: Comparison of the Effectiveness of Pedagogical Tools

rank these instruments on a scale 1-9 where 1 is the most effective instrument and 9 is the least effective instrument. We used a rank-order mechanism to rank the various instruments. The responses from students are shown in figures 1-4. To evaluate the responses and to make it easier for the readers to comprehend this data we have normalized the raw rank-order score on a scale of 100 points. Therefore, the cumulative score of all the instruments in any sub-figure in figures 2-4 is 100 points.

Overall in the class the *think-aloud* recitation sessions were the most effective teaching instrument with almost 17.4 points. Moreover, the think-aloud recitation sessions were the most effective learning instrument for male and female alike as shown in figure 3. In fact think-aloud recitation sessions were a more effective tool for female students with a score of 18.5. Furthermore, even the students who disagree or remained neutral to the problem ability statement rated the recitation sessions to be the most effective instruments of the course as shown in figure 4.

The next most effective instrument for the most part were the lectures. For both male and female students lectures were very effective. Students who agree and remained neutral also found the lectures to be effective. However, the students who disagree to the statement did not found the lectures to be as effective.

Google groups were also found out to be an effective tool especially for female students. During the ten weeks of the quarter there were 708 messages posted on the groups. This translates to 5.9 postings per student. Another thing that we noticed in particular was that the students who were otherwise shy to ask questions in class used to ask questions frequently on Google groups.

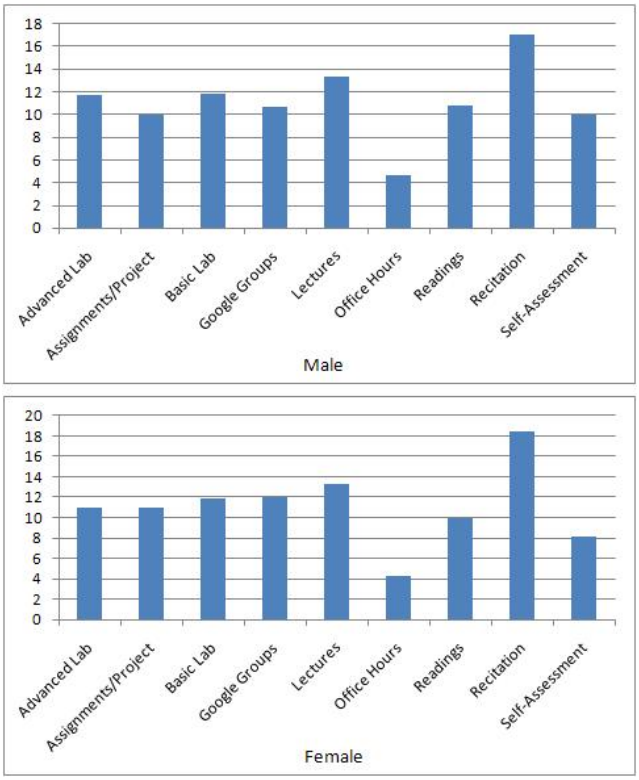

Figure 3: Comparison of Pedagogical Tools based on Gender

4. REFLECTIONS AND FUTURE WORK

The evaluations clearly show that think-aloud problem solving is a promising teaching tool in problem solving pedagogy. However, we would like to emphasize here that think-aloud problem solving is not an effective tool in isolation. Its effectiveness is leveraged by proper use of other teaching tools such as lectures , programming exercises and others.

It may seem that because think-alouds foster an imitation-based learning it hampers the students' ability to develop creative abilities. However, CS2 is just a start of a series of courses where student get enough practice of problem solving in diverse situations. Therefore, although a student is learning to imitate skills of someone else, but with time he or she will develop a unique skill of problem solving skills.

Think-aloud problem solving approach is a mean to an end but not an end in itself. This method of teaching is a prescriptive method as opposed to an investigative method. Therefore, our future work is to enhance the effectiveness

of think-alouds even further by looking at the students who were not able to learn effective problem solving in this course. Since the enrollment in CS2 classes is large an investigative approach is difficult to employ for all students. However, for students who have faced problems in learning we plan to use an integrated approach that includes both a prescriptive approach and an investigative approach. In investigative think-alouds we will be working on asking the students to do a think-aloud on a problem to see that at what point of problem solving are they having difficulty.

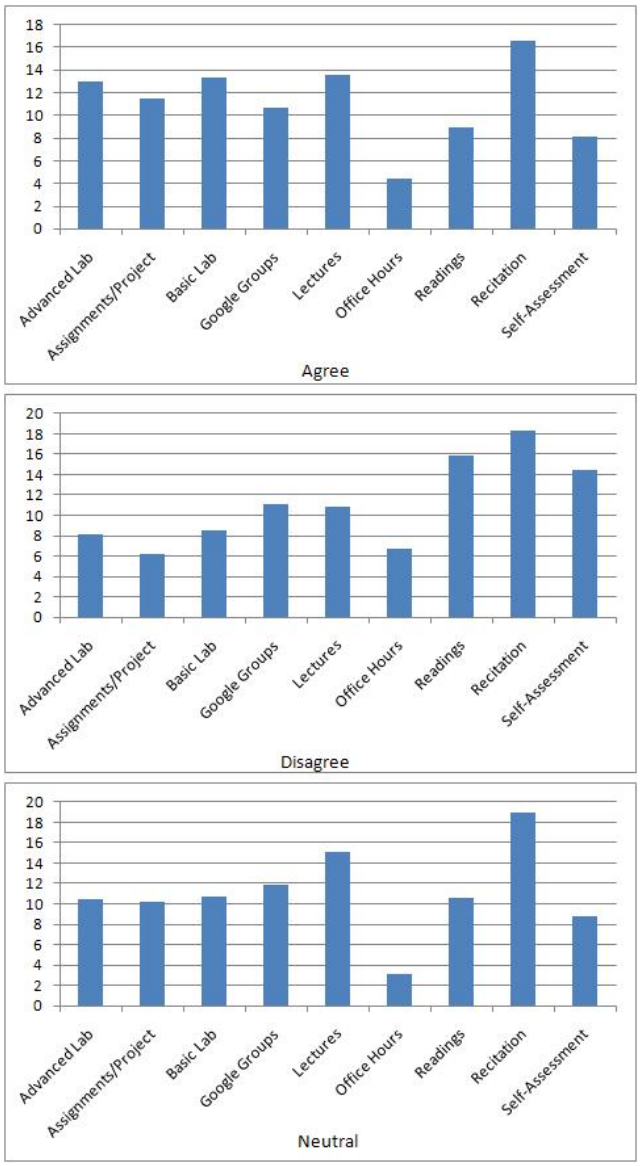

Figure 4: Comparison of Pedagogical Tools based on Programming Ability

5. ACKNOWLEDGMENTS

We would like to thank the TAs of the CS2 course: Ather Hameed, Ahmed Omer, Nava Zulfiqar, Abdul Basit and Amer Tahir for all their help in making this course a conducive learning experience for the students. We would also like to thank Professor Ashraf Iqbal for giving value feedback before and duringand after the course.

The material for this course has been developed in part by a grant from the National ICT RD Fund, Pakistan entitled "ICT Centric University Excellence Program".

6. REFERENCES

[1] Caspersen, M. E. *Educating Novices in The Skills of Programming*. PhD thesis, Department of Computer Science, University of Aarhus, 2007.

[2] David J. Barnes, Sally Fincher, and Simon Thompson. Introductory Problem Solving in Computer Science. In *5th Annual Conference on the Teaching of Computing* (Centre for Teaching Computing, Dublin City University, Dublin 9, Ireland, August 1997), G. Daughton and P. Magee, Eds., pp. 36–39.

[3] Ericsson, K. A., and Simon, H. A. Protocol analysis: Verbal reports as data, 1984.

[4] Fincher, S. What are we doing when we teach programming? *Frontiers in Education Conference, 1999. FIE '99. 29th Annual 1* (1999), 12A4/1–12A4/5 vol.1.

[5] Gries, D. A principled approach to teaching oo first. In *SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education* (New York, NY, USA, 2008), ACM, pp. 31–35.

[6] Guzdial, M., and Ericson, B. *Introduction to Computing and Programming with Java: A Multimedia Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[7] Howard, R. A., Carver, C. A., and Lane, W. D. Felder's learning styles, bloom's taxonomy, and the kolb learning cycle: tying it all together in the cs2 course. In *SIGCSE '96: Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education* (New York, NY, USA, 1996), ACM, pp. 227–231.

[8] Lahtinen, E., Ala-Mutka, K., and Jarvinen, H.-M. A study of the difficulties of novice programmers. In *ITiCSE '05* (New York, NY, USA, 2005), ACM Press, pp. 14–18.

[9] Robbins, K. A., Key, C. S., Dickinson, K., and Montgomery, J. Solving the cs1/cs2 lab dilemma: students as presenters in cs1/cs2 laboratories. In *SIGCSE '01: Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education* (New York, NY, USA, 2001), ACM, pp. 164–168.

[10] Robins, A., Rountree, J., and Rountree, N. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education 13* (June 2003), 137–172.

[11] Singer, J., Sim, S., and Lethbridge, C. *Software Engineering Data Collection for Field Studies*. Springer-Science, 2008, ch. 1, pp. 9–34.

[12] Walsh, L. N., Howard, R. G., and Bowe, B. Phenomenographic study of students' problem solving approaches in physics. *Phys. Rev. ST Phys. Educ. Res. 3*, 2 (Dec 2007), 020108.

[13] Winslow, L. E. Programming pedagogy – a psychological overview. *SIGCSE Bull. 28*, 3 (September 1996), 17–22.