A COLLABORATIVE APPROACH TO LEARNING PROGRAMMING: A HYBRID LEARNING MODEL

Linxiao Ma, John Ferguson, Marc Roper, John Wilson, Murray Wood
Department of Computer and Information Sciences
The University of Strathclyde, Glasgow G1 1XH, UK
Email: {Linxiao.Ma, jf, marc, jnw, murray }@cis.strath.ac.uk
URL: http://www.cis.strath.ac.uk/people/biography/{linxiao, jf, marc, jnw, murray}/

ABSTRACT

The use of cooperative working as a means of developing collaborative skills has been recognised as vital in programming education. This paper presents results obtained from preliminary work to investigate the effectiveness of Pair Programming as a collaborative learning strategy and also its value towards improving programming skills within the laboratory. The potential of Problem Based Learning as a means of further developing cooperative working skills along with problem solving skills is also examined and a hybrid model encompassing both strategies outlined.

Keywords

Collaborative Learning, Programming Learning, Pair Programming, Problem-Based Learning.

1. INTRODUCTION

Recent years have seen much discussion and concern regarding approaches to teaching and learning programming [1, 10, 11]. While a great deal of the debate has centered on the merits of an early use of an object orientated approach as opposed to a procedural approach [4] other concerns relate to the lack of early fostering of communication and collaboration skills between students; skills that are seen by industry as paramount and essential for team based software development. Many current approaches to teaching programming involve students spending much of their time alone with support restricted to demonstrator help during laboratory sessions. This approach is viewed by many as being inconsistent with a student's future professional life where they have to work with others [10]. It follows that a collaborative learning model would appear a more consistent approach in preparing students for their future career.

In the academic year 2003/2004, Pair Programming (PP) was employed as a collaborative learning model to support the introductory course "Programming Foundations" at the University of Strathclyde. Early results from this initiative look promising and show that PP has not only been well received by students, but overall student performance in laboratory work has improved. Current effort is focused on exploring the possible advantages of extending collaborative working to other areas of the learning process, in particular the potential of Problem Based Learning as a strategy for developing problem formulation and reflection skills.

2. PAIR PROGRAMMING

2.1 Background

Pair Programming refers to a form of collaborative working where two programmers work continuously on the same design, algorithm, code, or test. One of the programmers, the *driver*, controls the mouse and keyboard actively writing the code, while the *navigator* simultaneously reviews the work to detect mistakes and provide strategic suggestions [14]. It is an essential aspect of the approach that the two programmers reverse roles between driver and navigator after a designated time period; code written by only one member of the pair is not acceptable [9].

Following its wide acceptance in industry [12], the significance of PP within education has also been widely recognized. Firstly, it provides the potential for students to acquire knowledge through interaction with their partner [8]. Secondly, PP produces *pair pressure* between students helping students to focus on tasks and encouraging them to perform: most pair members are reluctant to disappoint their partner [13]. Thirdly, working in pairs helps ensure that students are more efficient at detecting and removing mistakes, reducing the student's sense of frustration in programming practice.

Some related research [9, 10, 14, 15] has been done to evaluate the effects of PP in education. In particular a number of experiments were carried out in the University of California Santa Cruz and in North Carolina State University to assess the efficacy of PP in introductory programming courses. The results from these experiments were encouraging and showed that students had a positive attitude toward PP and were more likely to continue study as Computer Science majors.

2.2 Pair Programming at Strathclyde

2.2.1 Course Setting

The "Programming Foundations" course at Strathclyde employs JAVA to provide students with a foundation in computer programming. The course consists of two one-hour lectures and one two-hour laboratory session each week. In the laboratory students complete a weekly assignment under the supervision of laboratory demonstrators. Students are assessed in mid-semester tests (20%) and in a final written examination (80%). In the academic year of 2003-2004, there were 274 students registered on this course, and while practical exercises were consistent with previous years, students were asked to work in pairs to complete weekly assignments.

At the end of the first semester an informal survey was carried out to assess the effects of PP. Data was collected in three ways: student test performances; questionnaires on students' experiences with PP; laboratory demonstrators' observations. In the following section we report the findings and discuss their implications.

2.2.2 Result & Discussion

Performance in Tests

Figure 1: Students' Performance in Test

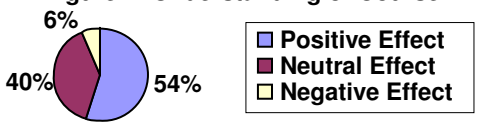
Two mid-term tests were performed in the first semester where students were asked to complete a small program that was evaluated on a 5-point scale. If a student completed all the functions correctly, they were awarded a score of "5". In the academic year of 2003-2004 when PP was employed more students obtained the highest score in both tests compared to the pervious year without PP, Figure 1.

Experiences with Pair Programming

At the end of the first semester students were asked to complete a questionnaire to gauge their experiences. The questionnaire covered the following areas:

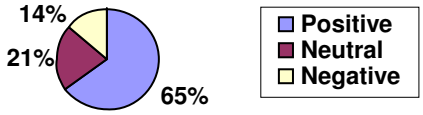
- Effect of PP on understanding course material;
- Helpfulness in laboratories;
- Enjoyment;
- Effect of PP on individual test performance;
- Pairing students with different abilities;
- Students' views on benefits and pitfalls of PP;
- Should it be used again in the second semester?

Figure 2: Understanding of Course



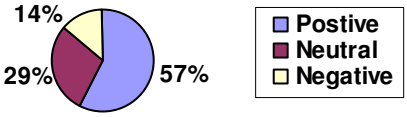
More than half of all students (54%) believed PP had a positive effect on their understanding of the course material.

Figure 3: Helpfulness in Laboratories



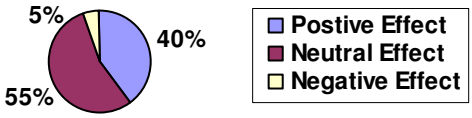
There were many more students (65%) who thought PP was helpful for them completing tasks.

Figure 4: Enjoyment



More than half of the students claimed the PP was enjoyable.

Figure 5: Individual Test Performance



40% of students thought PP had a positive effect, while only 5% claimed the opposite view. More students (55%) presented a neutral view toward to this question.

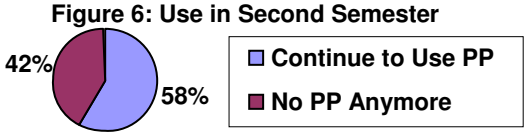
Most students (75%) liked to be paired with students of equal ability while a quarter of students preferred to work with students of perceived higher ability. Interestingly, there was only one student who wanted to be paired with a student of weaker ability.

The majority of students believed shared knowledge helped their understanding of the course contents. A few students thought PP was helpful for detecting and correcting mistakes, while only a small number claimed PP reduced the time taken. Additional benefits claimed by students included increased

confidence, less boredom, and a wider range of alternative solutions produced.

The main concern of PP was the imbalance in effort. Many students identified problems that occurred when one of the pair was not as well prepared as the other. Compatibility between members in the pair was also seen as a problem by many students. Some students claimed they could not understand the program written by their partner. In addition, some students thought it was difficult to get on with the other person. One possible reason for these problems was that many students did not have much prior experience or skills in communicating with their partner in a collaborative setting. A major benefit of PP is its ability to provide an opportunity for students to improve their communication experience and skills. This implies that compatibility may be viewed as a *problem* from a student's perspective, whereas it is in reality the *growing pains* of a necessary new skill, essential to their future careers.

Figure 6: Use in Second Semester



The majority of students (58%) claimed that they would like to continue to work in pairs in the following programming course. A higher than expected number of students (42%) had the opposite view. One possible reason for this was the perceived imbalance of contribution within pairs, leading to students who made a high contribution feeling that the activity was unfair.

Laboratory Demonstrators Observations

Some impressions on student performance in the laboratories were also collected through personal observations and feedback from laboratory demonstrators.

Firstly, it was noted that students were working more efficiently in their lab-based assignments and managing to complete the tasks quicker. Secondly, students were better prepared before arriving at the laboratory. Thirdly, there was increased communication between students, and between students and demonstrators with students far more likely to ask questions than in previous years. On the other hand, several problems were found with some students being overly reliant on their partner. Also, several students did not change roles between driver and navigator and some students worked alone, even when they had been assigned a partner.

2.2.3 Summary

Overall it would appear that PP has been accepted as an efficient learning method with, on the whole, a positive effect toward a student's experience in learning programming. However, the imbalance of contribution between members within pairs raises significant concerns that need to be addressed.

While Pair Programming holds promise for collaborative working and improving student programming skills, it has been recognized by the authors and by others [11] that students also exhibit difficulties engaging with material presented in lectures and transferring knowledge to practical problem solving. One approach that has received a great deal of attention in helping to overcome this difficulty is Problem Based Learning (PBL). In this model students are first challenged with a problem that motivates them to seek relevant knowledge and to integrate this knowledge into the problem solving process.

3. PROBLEM-BASED LEARNING

PBL has its roots in graduate medical teaching but has now been adopted in a wide range of educational domains, including computer science. Within PBL students acquire knowledge through solving *real-life* problems, working collaboratively in small groups. Detailed information on how to tackle a problem is often not provided directly to students, although resources are available to support them in formulating solutions [5, 6].

The skills developed by PBL are seen as important tools for life-long learning and essential to a student's future career. These can be summarized as [2]:

- developing reasoning and problem solving skills;
- promoting interpersonal skills and ability to work as team members;
- developing independent, self-directed critical thinking and learning skills.

PBL has received considerable attention as a learning strategy within computing education [6], with implementations in courses ranging from Web Site Development to Computer Networking [3]. Programming courses are also seen as a suitable domain for the adoption of PBL [7,11].

It is envisaged that PBL will provide an opportunity to deliver a range of materials in order to satisfy the demands of a diverse range of student abilities, ensuring that weaker students are not confused and stronger ones, bored. Furthermore, this shift of balance towards more practical work should improve the current situation where some students still lack confidence in their programming skills, even after 2 semesters.

3.1 Categories of PBL

Ellis and others [6] divide PBL approaches into three categories, namely *problem-based approach*, *guided problem-based learning*, and *full problem-based learning*. In the first category, lectures are

Figure 7: A progressive approach to PBL adoption

Novice Students	Intermediate Students	Advanced Students
The problem is simple and well-structured	The problem is complex and ill-structured	The problem is defined by students themselves.
Lectures present the conceptual knowledge	Students study the conceptual knowledge by themselves, but lectures establish the topics	Students are fully self-directed in their learning of conceptual knowledge
Students work in a group	Students work in a group	Students work in a group
The group activities are well- structured and well guided by lecturers	The group activities are designed and arranged by students themselves.	The group activities are designed and arranged by students themselves.
The Learning resources are well refined or chosen by lecturers.	Lectures suggest a set of possible learning resources to students. Students should pick up the relevant materials by themselves.	Students look for learning materials by themselves

employed to deliver the course material, supported by problems designed to engage students with the material. In guided problem-based learning some lectures are given to present only basic fundamental background knowledge and students then work in groups to solve problems. In addition, a range of resources are provided to allow students to acquire more detailed knowledge. Finally, in full problem-based learning, students work in groups and knowledge is no longer formally distributed directly by lectures, with the problem itself guiding and driving the entire learning process.

Selecting the appropriate PBL method will normally be based on the nature of the course and the ability of the students. In programming courses the approach taken will normally be based on the students' programming ability and problem solving experience. For the novice programmer, who may have come directly to university from a teacher-centered school environment, it would be considered by many to be a high risk strategy introducing them to programming through solving complex, ill-structured problems. At this stage, it would seem reasonable to challenge students with simple, well-structured problems, with the lecturer providing guidance on the course content along with the problem solving process. When students have accumulated sufficient well-developed, self-directed learning skills, the full problem-based learning approach would then be introduced.

Based on the second and third category of PBL methods outlined by Ellis, we have developed a three stage approach to PBL based on the ability of students, Figure (7).

For novice programmers, conceptual knowledge is presented in lectures and students work in groups with simple, well structured problems that cover the concepts presented in the lectures. When students have accumulated sufficient experience of problem solving and programming they can progress to the intermediate model where problems are more complex, ill-structured and there is no formal exposition of conceptual knowledge from lecturers. At this stage students have greater control over group activities and learning resources. In the final approach students are presented with greater challenges and have more freedom in controlling the problem solving process. Students define the problem within the context of course topics, and are fully self-directed in group working and information gathering.

3.2 A Hybrid Approach

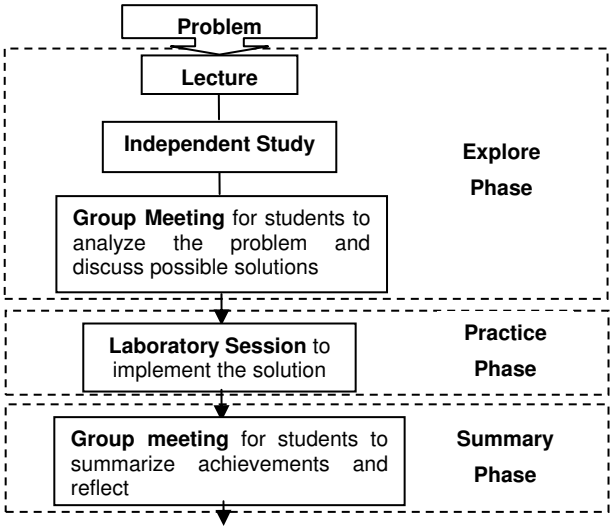
The programming learning process can be divided into three phases, namely the *explore* phase, the *practice* phase, and the *summary* phase, Figure 8. In the explore phase, students analyze the problem, absorb the relevant knowledge, such as basic concepts, essential library classes and relevant algorithms, then discuss possible solutions. The practice phase is where students obtain their programming experience and skills through implementing their solution. Finally, in the summary phase, students analyze their achievements and review any problems they encountered.

It is apparent that as a group-based learning model, PBL is beneficial for both the explore and the summary phases. However, there is no substantial evidence to support PBL as an efficient way of optimizing a student's practice in programming and it is here that PP provides a pivotal role, with the combination of PBL and PP expected to provide comprehensive support for learning.

Compared to the traditional first year programming course, the proposed restructured course at Strathclyde has three main differences. Firstly, students will work in groups, or clusters of the order of three pairs, to solve problems in each learning unit, rather than just carrying out laboratory based work individually. Each learning unit begins with a problem and the entire learning process is problem-driven and the role of lectures is to present the fundamental concepts that are useful for developing a solution to the problem. The second difference is that there will be two additional group meetings for students; one group meeting is used by students to analyze the problem and discuss possible solutions; the other group meeting allows students to summarize their achievements and reflect on any problems encountered. The final difference is the

continued use of PP in the laboratory work to optimize their programming practice.

Figure 8: Learning Lifecycle Model



4. CONCLUSION

The development of collaborative skills has been recognized as imperative in programming education. In this paper we have presented results from preliminary work to implement PP as a collaborative learning model within our introductory first year course "Programming Foundations". Early results look promising and show that PP has potential within the context of an introductory programming course to enhance the development of both programming practice and collaborative skills. These early indicators have provided impetus for us to expand the benefits of collaborative learning to seek solutions for promoting problem solving skills. Problem-based learning offers a potential solution to this difficulty.

In this paper a hybrid model that integrates Pair Programming and Problem-Based Learning is presented. This hybrid model is expected to provide a collaborative learning framework where students can harness the benefits of PBL when they study conceptual knowledge as well as utilize the advantages of PP when they apply this knowledge.

5. REFERENCES

[1] Barg M., Fekete A., Greening T., Hollands O., Kay J., Kingston J.H., *Problem-based learning for foundation computer science courses*, Computer Science Education 10 (2000), 1--20.

[2] Barrows H., *Problem-based, self-directed learning*, the Journal of the American Medical Association (JAMA), Vol.250, (1983).

[3] Beaumont C., Sackville A., *Cheng C., Identifying Good Practice in the Use of PBL to teach Computing*. ITALICS e-journal, Vol.3 Issue.2 (2004).

[4] Bruce K., "Controversy on How to Teach CS 1: A Discussion on the SIGCSE-members Mailing List", SIGCSE Bulletin (2004).

[5] Boud D., Feletti G., *The Challenge of Problem Based Learning. Kogan Page, London*. (1991)

[6] Ellis A., Carswell L., Bernat A., Deveaux D., Frison P., Meisalo V., Meyer J., Nulden U., Rugelj J., Tarhio J., *Resources, tools, and techniques for problem based learning in computing*. In Work Group report of the 3rd annual SIGCSE/SIGCUE ITiCSE conference (1998).

[7] Fekete A., Greening T., Kingston J., *Conveying technical content in a curriculum using Problem-Based Learning*. Proceedings of the Third Australasian Conference of Computer Science Education. ACM Press: Brisbane. (1998)

[8] Flor N., Hutchins N., *Analyzing Distributed Cognition in Software Teams: A Case Study of Team Programming During Perfective Software Maintenance*, presented at Empirical Studies of Programmers: Fourth Workshop (1991).

[9] McDowell C., Werner L., Bullock H., Fernald J., *The Impact of Pair Programming on Student Performance, Perception and Persistence.* Proceedings of the 25th International Conference on Software Engineering (2003).

[10] Nagappan N., Williams L., Ferzli M., Yang K., Wiebe E., Miller C., Balik S., *Improving the CS1 Experience with Pair Programming*. Presented as SIGCSE (2003).

[11] O'Kelly J., Bergin S., Gaughran P., Dunne S., Ghent J., Mooney A., *Initial finding on the impact of an alternative approach to Problem Based Learning in Computer Science*, present at Pleasure By Learning (PBL) conference, Cancun, Mexico (2004).

[12] Williams L., *The Collaborative Software Process*. PhD. Dissertation. University of Utah, Salt Lake City (2000).

[13] Williams L., Kessler R., *The Effects of "Pair-Pressure" and "Pair-Learning" on Software Engineering Education*, in Thirteen Conference on Software Engineering Education and Trainning. Austin Texas: IEEE Computer Soc (2000).

[14] Williams L., McDowell C., Nagappan N., Fernald J., Werner L., *Building Pair Programming Knowledge through a Family of Experiments*, Proceedings ISESE (2003)

[15] Williams L., Yang K., Wiebe E., Ferzli M., Miller C., *Pair Programming in an introductory Computer Science Course: Initial Results and Recommendations.* Presented at OOPSLA Educatior's Symposium, Seattle, WA (2002)