2006

Student Perceptions of Problem Solving through a Pair Programming Technique

J E. Terry
Edith Cowan University

P A. Williams
Edith Cowan University

R J. Mahnckel
Edith Cowan University

Terry, J.E., Williams, P.A.H. and Mahnckel, R.J. Edith Cowan University, Australia. Student Perceptions of Problem Solving through a Pair Programming Technique

Terry, J.E[1]., Williams, P.A.H[2]. and Mahnckel, R.J.

School of Computer and Information Science
Edith Cowan University, Australia
Email: j.terry@ecu.edu.au
Email: trish.williams@ecu.edu.au
Email: r.mahncke@ecu.edu.au

ABSTRACT

Research suggests that it is important to facilitate interaction between students as well as engagement with course materials in the first year of university. In addition, there is increasing emphasis on graduate abilities in teamwork, communication and problem solving. In the software development industry, teamwork is essential. One innovative methodology for modern software development is paired programming, a technique that has not been widely addressed at the tertiary level. This research evaluates the success of implementing a paired programming technique with first year computer science students, through the evaluation of the learning experience from the students' perspective.

INTRODUCTION

The traditional model of teaching aims at imparting information and transmitting structured knowledge, whilst the student-centred approach aims to facilitate understanding and promote conceptual change and intellectual development (Kember, 1998, p.23). As such teaching strategies must come second to student learning facilitation for effective student outcomes to be met (Ramsden, 2003, pp.145-147). Student-centred approaches are characterized by interaction and active engagement, and in teaching computer programming this is essential. One method to achieve this is pair programming. Pair programming is one approach to the design, development and testing of computer programs in which two programmers work together as a team sharing one computer. It is a quite different approach to an activity that has traditionally been solitary in nature (Object Monitor, 2001).

This paper discusses the success of using the pair programming method from the students' perspective. The method was employed with all first year computer science students enrolled in a foundation programming unit, as part of the assessment requirements. This paper reports the findings of a student learning experience survey conducted at the conclusion of the semester in which the pair programming method was introduced.

PAIR PROGRAMMING

The quality crisis in software development has been met with a number of methodologies addressing management, quality and productivity issues directly for example the Spiral, Agile and Extreme Programming methodologies. The Extreme Programming (XP) methodology has specifically incorporated the technique of pair programming into the costly coding phase, in the belief that 'two heads are better than one'.

Essentially one of the team, known as the driver, occupies the computer doing the keying function. The other is able to consider the problem space without that distraction, including reviewing resources, scribbling down design ideas and considering test data. Each team member has equivalent time being the driver. At first view the technique would appear to utilise two resources to achieve what would customarily be done by one. It is an essential ingredient of the methodology that all code is developed in this way, as opposed to two programmers individually developing parts of the final program and then bringing them together. Problem understanding, design issues, solution structure are all enhanced by the team approach.

The acceptance of pair programming requires a paradigm shift from IT management who could imagine productivity decreasing by 50%. Williams and Upchurch (2001) found that student paired programmers are only 15% slower than two independent individual programmers, but produce 15% fewer code errors. Research by McDowell, Werner, Bullock, & Fernald (2006, p. 90) found that “the pairing students produce higher quality programs, are more confident in their work, and enjoy it more”. Further the enjoyment of the pair programming experience as shown in Figure 1, was measured by Cockburn and Williams (2000) over a number of semesters and they found that students on average enjoyed the programming experience 85% more than programming alone.

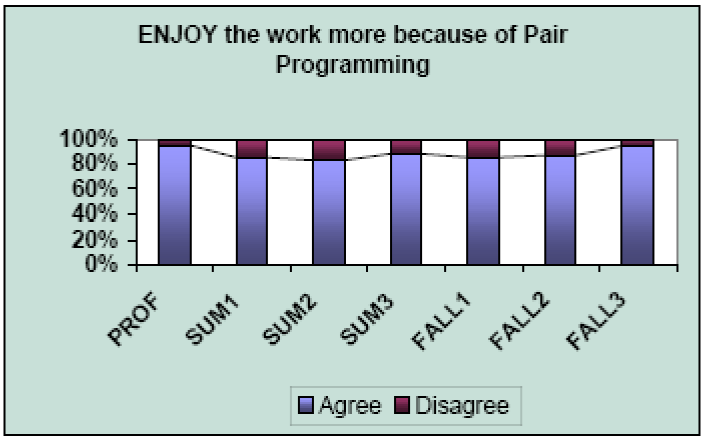

Figure 5: Pair Enjoyment (Cockburn & Williams, 2000)

Muller (2005) has shown that pair programmers are as cost effective as solo programmers if producing code with fewer errors. In this sense “programmer pairs and single developers become interchangeable in terms of development cost” (Muller, 2005, p. 166).

As with many methods there are drawbacks. Whilst some drawbacks are inherent on the method, others can be moderated depending on the environment, for instance, where more experienced programmers may find it tedious to work with a novice, in the educational environment most students are at a similar developmental level. Also, if attempting to measure productivity then team assessment can be problematic in both a work and educational situation. Whilst issues of programming style may be present, in the educational setting students are encouraged to work through the differences to be able to meet the assessment challenge.

Transforming student learning

According to Lui & Chan (2006) pair programming promotes not only quality programming skills, but responsibility, mentoring, teamwork and increased enjoyment. This research is confirmed by Williams and Upchurch (2001) who report that this technique works will to produce higher quality results particularly in student work. It is becoming increasingly important in tertiary education to foster lifelong learning skills and promote key skills in graduates (Sumsion & Goodfellow, 2004). Such skills are collectively referred to as graduate attributes and include problem solving, teamwork and workplace applied competencies (ECU, 2002). Pair programming clearly addresses these key skills. Further, the economic pressures of tertiary education mean larger class sizes which is accompanied by evidence that larger class sizes limit the opportunity for higher-level student learning (Karakaya, Ainscough, & Chopoorian, 2001). Pair programming may alleviate some of the workload placed on teachers in large classes as it promotes shared learning between students and therefore improved problem solving. Increased student motivation is also achieved as the method incorporates active participation, clear expectations, formative learning and supportive feedback (Biggs, 1999; Gross-Davis, 1993).

METHODOLOGY

The subjects were first year students at Edith Cowan University in Perth, studying an introductory programming unit, 'Programming Principles', over a thirteen week semester. A main objective of this unit was to allow the students to start developing the skill set required by a commercial programmer. Initially, students undertook an individual programming assignment over four weeks. Subsequent to this, students chose a partner to complete the remainder of the semesters' work with. Instruction was provided on the pair programming method. Students completed four programming tasks in pairs after which the learning experience survey was administered. One survey was completed per team, requiring students to agree on a response as a team.

The survey consisted of twenty questions asking respondents to compare the pair programming activity to programming alone. The questions were grouped into the following areas: enjoyment, problem solving, error detection, team work, technique, time management, and overall satisfaction.

A limitation of the research methodology would be that through luck or good management, some pairs gelled and enjoyed working as a team, whereas others resented the concept and would have preferred working alone or were with a partner who felt this way. This would have affected perceptions of the process.

RESULTS

Fifty teams responded to the survey. The results presented reflect the survey question groupings.

Enjoyment

In relation to the satisfaction of pair programming compared to programming alone (figure 2), the majority of students (78%) felt their experience was positive. It was also observed that when pair programming technique was introduced the students reacted positively by engaging with each other and creating an identity for their team by selecting a team name. It was also observed that a small number of teams did not interact as well as other teams.

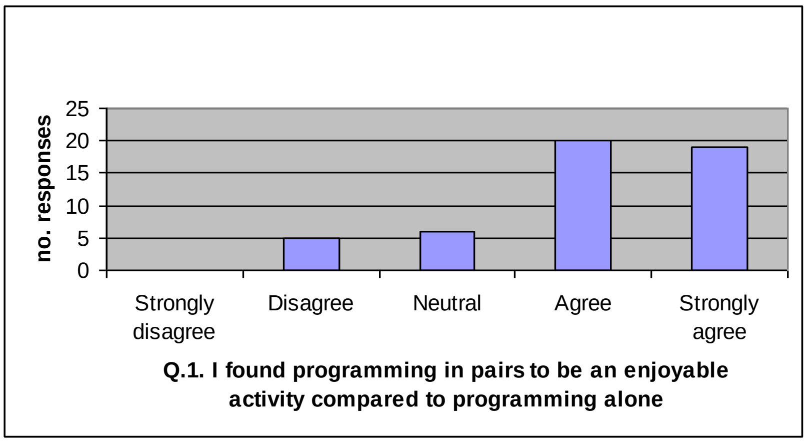

Figure 2: Perception of Enjoyment

Problem solving

The survey included three questions regarding the ability to problem solve as a pair compared to individually. 82% of respondents considered that the quality of the solution was better than could have been achieved by themselves. Additionally, the perception of the usefulness of pair programming was assessed by respondents' relation to four distinct programming tasks (Table 1).

Table 1: Usefulness of programming tasks.

Task	Usefulness
problem understanding	78%
solution design	72%
coding	76%
testing	80%

Figure 3 indicates that an 82% agreement in the ability of pair programming to aid exploration of potential solutions. This is consistent with the literature citing pairs investigating more design options than individuals.

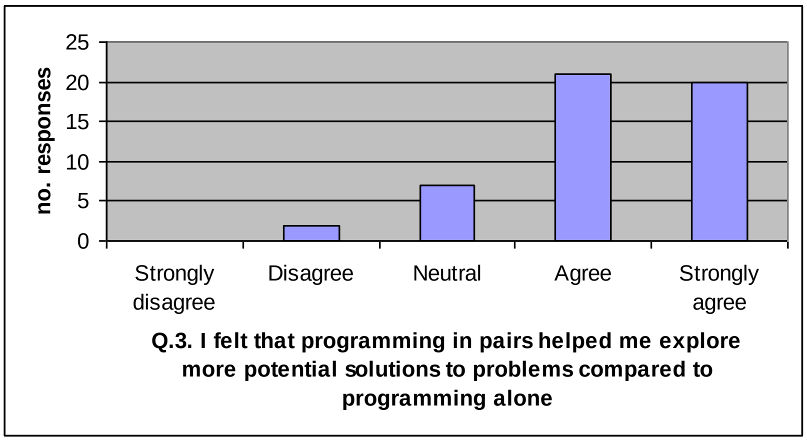

Figure 3: Exploration of potential solutions.

Error detection

The two questions of error detection related to the ability of pairs to detect more errors and to detect errors faster than individuals.

76% agreed that the pair detected errors faster than individuals, and as Figure 4 shows, 78% agreed that the pair detected more errors. The questions do not specify at which point in the program development the errors were detected. Problems could be uncovered anywhere in the range of activities they were expected to cover: problem understanding, program design, coding and testing.

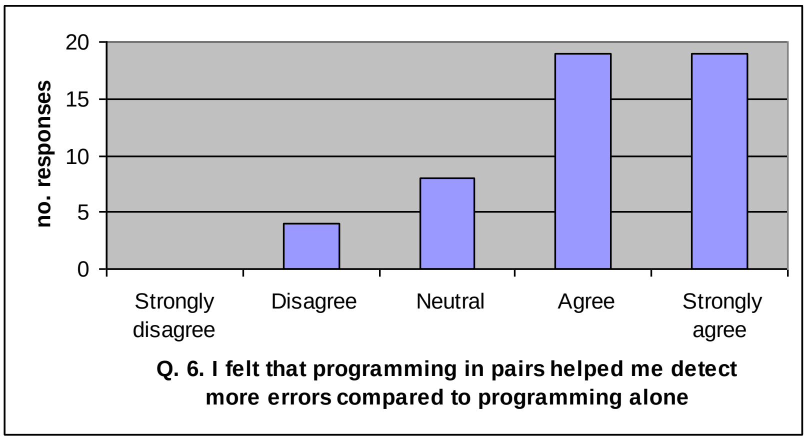

Figure 4: Error detection

Team work

The questions in this grouping asked the programming pair such questions as if they felt

- they continuously collaborated,
- focussed their efforts continuously,
- they contributed equally,

- more pressure to succeed.

68% of respondents felt that they had contributed to the task equally, however only 54% of respondents thought that pair programming had helped them continuously focus their efforts on the programming activity when compared to programming alone.

Figure 5 shows that 64% felt more pressure to succeed when working as a pair. This is an indication that for most, the team was greater than the sum of its parts in the desire to do well. This question had a high number of neutral responses with 34%, indicating almost no disagreement to the proposition.

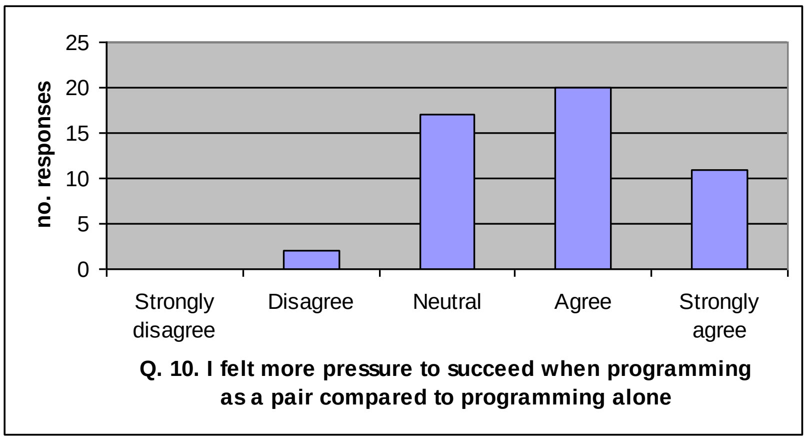

Figure 5: Pressure to succeed.

Technique

In this section of the survey students were asked about their contribution to the task when not controlling the entry of code at the keyboard. Figure 6 indicates that 62% of respondents felt that they were able to contribute fully when not in the driver role. A previous question supported this finding by indicating that only 16% of respondents felt they were easily distracted when not the driver.

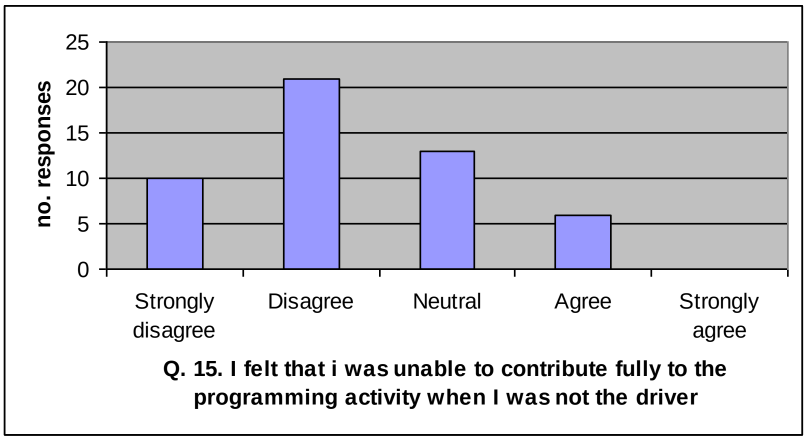

Figure 6: Contribution to Programming Activity

Time management

Figure 7 indicates that 48% of respondents felt they would have spent more time if they had completed the work alone. Therefore, respondents felt that there were time benefits in doing the work as a team.

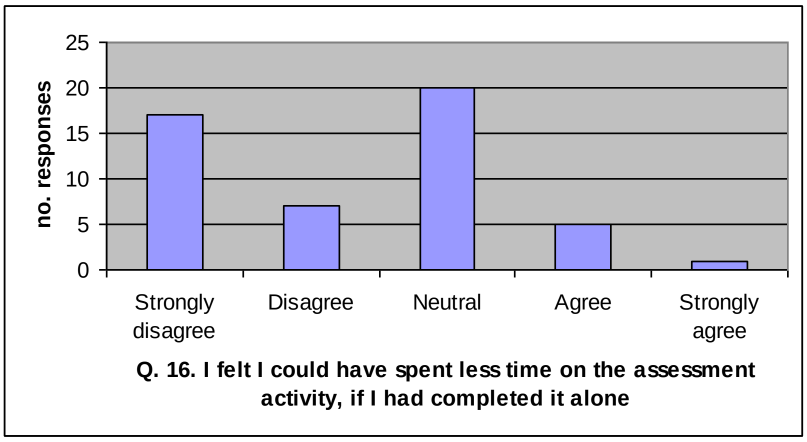

Figure 7: Time management

Overall satisfaction

The results of overall satisfaction with the technique (figure 8) show that 74% of respondents believe that pair programming enhances their programming skills. Further, that 70% of respondents indicated that they would rather program in pairs than alone.

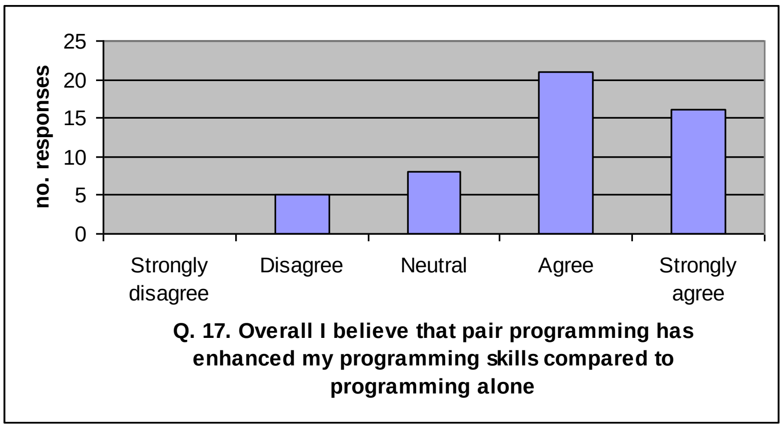

Figure 8: Overall satisfaction

DISCUSSION

The indications are that the use of the pair programming technique has been a positive learning experience for student in this research group. The enjoyment factor reflects this. Despite these positive results it should be acknowledged that in some cases the respondents did not feel they worked well in a team. This may be attributed to various factors including:

- some students are not so team-centric and prefer to work individually,
- part-time students find it more difficult to arrange times to work in a team environment,
- students are in a pair with someone they are not comfortable working with.

As was found in the literature, students felt they were able to find more errors in a shorter time in a team than they would individually. In addition, the results indicate that the student reflection on their own abilities in problem solving was enhanced by working in a team. However, it is not surprising that when two people are working on a problem that there are times when one partner may be less focus or distracted, although the results show that balanced contribution was not an issue for the majority of teams, Despite this, the technique did not necessarily affect their focus on the task, whilst the majority of students indicated that working as a team did foster a desire to do better than working alone.

In pair programming, the non-driver does not touch the keyboard at all and could feel that they are not making an equal contribution. This may well be the case if the same person was always the driver. In the application of the technique in this research the student were required to rotate this role to ensure both parties experienced the hands-on as well as the ‘thinking’ role, and therefore appreciated the value and responsibility of each role in the team. These research results indicate that whilst most students felt they contributed, not all students felt they this was a significant contribution when they were the non-driver.

In terms of time management and the associated effort required, the survey results indicate that there was a clear perception that there was no extra time burden to working as a pair. However this question does not address the extent to which the time taken was less as a pair. Half of the teams considered it profitable in terms of time when working as a team. It should be noted that time management is a crude measure for productivity which looks at output in a time period, it does not consider the quality of this output.

The results for overall satisfaction show that the majority of students thought their programming skills were enhanced by the pair programming technique and that they preferred it to programming tasks attempted alone. Again this is consistent with other research undertaken with students at the tertiary level.

CONCLUSION

This research is the initial step in the investigation of this educational technique to assist students prepare for post-degree work. The pair programming technique as well as being a novel approach being used in industry, is an innovative learning technique for teaching programming to tertiary level students. These results can be extended by correlating the student experience with actual student results.

Overall the responses indicate a positive attitude to the pair programming experience and its role enabling each team member to participate productively. This research confirms that most students feel they had gained useful experience and improved their skills by using this technique. The introduction of pair programming in this unit appears to have enhanced the programming skills of these students. This is a very positive outcome for those using or considering using this technique for teaching programming to undergraduates.

REFERENCES

Biggs, J. (1999). Teaching for Quality Learning at University. SRHE and Open University Press, Buckingham, UK.

Cockburn, A. and Williams, L. (2000). The costs and benefits of pair programming, URL http://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF, [Accessed 20 September 2006].

ECU. (2002). Graduate attributes @ ECU (Handout). Mount Lawley, WA: Centre for Learning and Development Services, Edith Cowan University.

Gross-Davis, B. (1993). Motivating students. In B. Gross-Davis (Ed.) Tools for Teaching, Jossey-Bass Publishers, San Francisco.

Karakaya, F., Ainscough, T., J. and Chopoorian, J. (2001). The Effects of Class Size and Learning Style on Student Performance in a Multimedia-Based Marketing Course. Journal of Marketing Education, 23(2): 84-90.

Kember, D. (1998). Teaching Beliefs and Their Impact on Students' Approach to Learning. In B. Dart & G. Boulton-Lewis (Eds.), Teaching and Learning in Higher Education (pp. 1-25). ACER Press, Melbourne.

Lui, K. M. and Chan, K. C. C. (2006). Pair Programming Productivity: Novice-Novice vs. Expert-Expert. International Journal of Human-Computer Studies, 64(9): 915-925.

McDowell, C., Werner, L., Bullock, H.E. and Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. Communications of the ACM. 49(8): 90. Association for Computing Machinery.

Muller, M M. (2005). Two controlled experiments concerning the comparison of pair programming to peer review, The Journal of Systems and Software,78(2): 166-179.

Object Monitor. (2001). PairProgramming.com. URL http://www.pairprogramming.com/ [Accessed 20 September 2006].

Ramsden, P. (2003). Learning to Teach in Higher Education (2nd ed.). RoutledgeFalmer, London.
Sumsion, J. and Goodfellow, J. (2004). Identifying Generic Skills through Curriculum Mapping: a Critical Evaluation. Higher Education Research and Development, 23(3): 329-346.

Williams, L. and Upchurch, R. L. (2001). In support of student pair-programming. URL http://collaboration.csc.ncsu.edu/laurie/Papers/WilliamsUpchurch.pdf [Accessed 20 September 2006].