Routledge
Taylor & Francis Group

Innovation in Teaching and Learning in Information and Computer Sciences

ISSN: (Print) 1473-7507 (Online) Journal homepage: http://www.tandfonline.com/loi/rhep14

Embedding Inquiry based learning into Programming via Paired Assessment

Sonya A. Coleman & Eric Nichols

To cite this article: Sonya A. Coleman & Eric Nichols (2011) Embedding Inquiry based learning into Programming via Paired Assessment, Innovation in Teaching and Learning in Information and Computer Sciences, 10:1, 72-77

To link to this article: http://dx.doi.org/10.11120/ital.2011.10010072

Embedding Inquiry based learning into Programming via Paired Assessment

Sonya A. Coleman
School of Computing and Intelligent Systems,
University of Ulster,
Northland Road, Londonderry,
BT48 7JL, N. Ireland
sa.coleman@ulster.ac.uk

Eric Nichols
School of Computing and Intelligent Systems,
University of Ulster,
Northland Road, Londonderry,
BT48 7JL, N. Ireland
nichols-e@email.ulster.ac.uk

ABSTRACT

Changes within our approach to teaching can make some students feel uncomfortable. To overcome this, inquiry based learning, which is strongly supported by research in the areas of intellectual development and approaches to learning (Prince, 2007), can be used. Inquiry based approaches should be introduced in combination with existing teaching styles in order to address the needs of all students. Pair programming enhances the communication among peers and encourages students to ask questions of each other and be more ambitious in their computer programming practicals. The students subsequently gain confidence from one another to try different approaches to solving programming problems; this enhances deeper learning. Additionally, working in pairs provides some students with the courage to ask questions of the teacher while with their pair, which they may not do alone. This paper presents a case study on using pair programming to encourage inquiry based learning within programming modules, to improve attendance and practical assessment results.

Keywords

Pair-programming; assessment

1. INTRODUCTION

The concept of pair-programming has recently been introduced in U.S. universities and companies. By definition, pair programming is a technique in which two programmers work together on one task at one PC. One programmer does the coding (the driver) while the other programmer reviews each line of code as it is typed in (the observer); the two programmers frequently switch roles. The observer should also be coming up with ideas for code improvements allowing the driver to focus all of their attention on the tactical aspects of completing the current task (Williams). Within industry, pair programming improves many aspects of development including design quality, reduced defects, enhanced technical skills and improved team communications (Cockburn 2001). These are skills that we also require in our students. In 2000 (in Missouri State University) students were asked their views of pair programming, 70% of them favoured it (Sanders, 2003). Much research has been carried out and published on the benefits of pair programming within both industry and academia. For example, within an industrial setting, Arisholm et al., (2007) carried out a controlled experiment in pair programming using a combination of junior, intermediate and senior Java consultants within Europe. The results suggested that the most benefit was gained by the junior developers in that they could complete tasks in less time and more successfully than normal with the support of a partner. The research in (Dyba, 2007) indicated that, given the correct pairing, pair programming could be very successful. Hence in this case study the students are permitted to select their own partners with the aim of finding the right pairs. Many projects have been undertaken in pair programming in academia, but primarily in the U.S. rather than the U.K. The work of Braught et al., (2008) determined that students are more likely to successfully complete their course using pair programming and are also strengthened as individual programmers. In (McDowell, 2006; Simon, 2008) this was also found to be true as students have more confidence in their work and particularly in the case of females in computer science.

Much research has also been carried out and published on the benefits of both inquiry based learning and working in small groups. Inquiry based learning is an approach to increasing retention through the use of a teaching approach that allows student-constructed learning rather than teacher transmitted information (The Sparks Foundation, 2010). An old adage states: "Tell me and I forget, show me and I remember, involve me and I understand." This is the basic principle behind inquiry based learning: it is any task or activity that

involves the students to get them thinking for themselves, discussing problems together and ultimately embedding deep learning. Therefore by introducing pair-programming, we are encouraging the students to work together to discuss and solve problems with the aim that if they are involved in creating solutions then they will have a better understanding of programming. It has many benefits including increasing student involvement, encouraging collaboration among students and developing a deeper knowledge due to the students thinking through the processes themselves (Prince 2007; Prince 2006). The key aspects of inquiry based learning from the students point of view are that the students: are involved in the process of learning; engage in an exploration process; raise questions, propose explanations, and use observations; plan and carry out learning activities; communicate with staff and peers (Prince 2007). Ideally in any computer science or engineering programme we aim to provide our students with an appropriate level of theory, combined with practical applications that enable the students to investigate the theory for themselves, develop it and understand it. Within tertiary education we all aim to have our modules as interactive as possible. However, we need to be careful in getting the balance correct: it is a well known fact that different students learn in different ways; some learning from doing, others from reading, some even learn better with background music. Therefore significant changes within our approach to teaching could make some students feel uncomfortable, and hence in this paper we have introduced inquiry based combination with existing teaching styles in order to address the needs of all students.

In this paper we present initial findings on the use of pair-programming as a means of embedding enquiry based learning activities into an algorithmic programming module. We find that the students enjoy working in pairs and it provides an overall improvement in the module assessment marks.

2. METHODOLOGY

Pair programming as a means of inquiry based learning was introduced into the Algorithmic Programming II module in semester 2 of the academic year 2009-10 (AY0910). There is a wide range of first year students who take this module including BSc Computer Science, BEng electronics and Computer Systems and BEng Games Development. With respect to carrying out practicals and practical assessments, the students were put into pairs.

2.1 Pairing

In total there were 58 students on this module. The students were allowed to choose their own partners, as randomly selecting partners can lead to many problems during the course of the module. In week 1 the students were ask to email the author with details of the partnership (one email per pair). There were initially some problems with this. Some students selected their partner without asking them if they were willing to be their partner and this resulted in some students being in two pairs. These issues were raised in the lecture and students were asked to solve it among themselves. Some students were also reluctant to work in pairs, but it was not optional, so they had to commit to working with someone. Of course, by week 3, a few students had dropped out and so their partners had to be paired with someone else. However, none of these issues were major, and were easily resolved prior to the first assessment taking place.

2.2 Assessment

In their pairs, the students completed the practicals and were assessed. However, assessments were still conducted on an individual basis in the following manner:

a) both students had to be present at assessments in order to obtain their individual mark;
b) both students were asked individual questions about the practical, hence both students had to demonstrate their own understanding.

By conducting the assessment in this way, no individual student received credit for their partner's work.

The assessments were conducting during the timetables practical time. Approximately 5 to 10 minutes was spent with any one pair and so they were able to return to their normal practical work. At the beginning of the practical class, all students were requested to select which PC the practical would be running on and have it running ready for the assessment to be conducted. The practical class would be commenced as normal and then the assessment would begin. The assessor simply asked each pair to run their program, demonstrate their output and then show their code.

Each student in the pair was also asked individual questions to gauge whether each student understood the code or if one had essentially done the work. Each assessment was marked out of 10, with 2 marks given for each of the following:

1. program running;
2. correct output;
3. good code structure;
4. good commenting;
5. understanding.

Obviously, the marks allocated for 1 – 4 were the same of each students, but an individual mark was then allocated for understanding. This was adequate over the five assessments to ensure individual marks.

At the end of the module, the success of pair programming was assessed in two ways: a statistical analysis was conducted to assess the overall improvement of coursework marks and attendance at the practical sessions in contrast to the previous year; the students were given a questionnaire in order to collect and analyze their views on pair programming. As this project was carried out in second semester, the first year students had already completed one programming module in first semester without using pair programming, so they were able to compare and contrast the advantages and disadvantages of both learning styles.

It should be noted that practical assessment was only one component of the coursework for this module. There were two class tests which each student took independently and these were combined with the paired assessments marks to create a individual assessment mark for each student.

In addition, after the exam component of the module was completed, additional statistical analysis was carried out to ascertain if the overall exam performance has improved in comparison to the previous year, thus providing evidence that pair-programming leads to deeper learning and understanding. However, this will also have to be cross correlated with other module marks to determine if an overall improvement is unique only to the Algorithmic Programming module, or if indeed the student cohort is academically superior to the previous year.

3. STUDENT VIEWS

Question	% response of 'yes'
Do you enjoy working in pairs?	76.2
Would you have preferred to work on your own?	28.6
Would you have preferred to work in larger groups?	28.6
Do you feel working in pairs has helped your understanding of programming?	57.1
Do you feel you have learnt from your programming partner?	57.1
Do you feel your programming partner has learnt from you?	42.9
Did you and your partner work together on programming practicals outside the assigned time?	47.6

Table 1. Summary of student feedback

At the end of the module, the students were issued with a questionnaire in order to obtain their views and opinions on pair programming. They were asked to compare and contrast the pair programming technique used in this project with the solo programming approach used in their first semester programming module. This provided us with the student's overall view of pair programming. In total, 21 students completed the questionnaire out of a possible 58. The questions and the percentage of students that answered 'yes' to each question is summarised in Table 1.

As illustrated in Table 1, 76.2% of the student cohort enjoyed working in pairs. The general feedback suggests that knowledge transfer within the partnerships did occur. The evidence also indicates that the students prefer working in pairs rather than larger groups which could potentially have impact on many other modules that incorporate group work as means of assessment. It is also interesting to note that 47.6% of the student cohort worked on their practicals together outside of the timetabled practicals; perhaps the use of pair programming helps to strengthen bonds and friendships amongst first year students.

4. EVALUATION

4.1 Quantitative Evaluation

The quantitative results of embedding pair programming into the Algorithmic Programming II module are presented. In order to conduct statistical analysis thoroughly, we firstly compared the attendance with that of the Algorithmic Programming II module in AY0809 to ascertain if attendance had improved. In addition, to ensure that any improvement in attendance was not dependent on the specific cohort, we analysis the attendance of a Mathematics module in semester one of AY0910 to obtain expected attendance profiles. Details of how the analysis was conducted are in Appendix A.

An overall aim of embedding pair programming as a means of enquiry based learning in the programming module was to improve practical attendance, enhance deep learning and thus improve practical assessment marks with the ultimate goal of improving first year retention. With respect to practical attendance, we found that there was an overall increase of 2.5% in practical attendance in comparison to the AY0809. Additionally, when we cross-correlated the second semester data with first semester data for the same cohort we found that the attendance was significantly higher than expected: practical attendance was 27.8% better than expected. We investigated whether working in pairs during the practical was having any impact on the attendance at lectures and found lecture attendance was 10.35% better than expected. This implies that working in pairs encouraged students to attend in second semester, however there is still room for significant improvement in the overall attendance.

We subsequently analysed the overall practical assessments for the module to determine if pair programming improved the practical assessment results. Analysis showed the practical assessment marks had increased by 31.2% in comparison to last year's practical assessment marks and were 26.2% higher than the expectation. Cross-correlation with other modules has also been conducted which highlights that attendance and performance in the Algorithmic Programming module is slightly better than the Mathematics module. However, it is not yet clear whether this is strictly due to the pair-programming or the students' module preference and hence next year we will try to gauge this via the student questionnaire by including questions which will advice us about which modules students like and dislike.

In addition to the analysis conducted, we analysed the attendance and performance of the students we deemed to have engaged in the module, i.e., those that submitted all of their coursework and completed the exam. If we look only at these students and compare their performance in Algorithmic Programming II with that of other modules which the same student cohort has completed, we find a significant improvement in both attendance and assessment as demonstrated in Table 1.

	% improvement
Practical attendance – all students	+20.6%
Practical attendance – females	+21.8%
Lecture attendance – all students	+19.9%
Lecture attendance – females	+15.7%
Assessment marks – all students	+53.5%
Assessment marks – females	+61.0%

Table 1 – Summary of cross correlation with the mathematics module

4.2 Qualitative Evaluation

From the questionnaire feedback and general discussion with the students, it was found that they enjoyed the pair programming aspect of the module. Not only did they conduct their practicals in pairs but this also extended to going to lectures together and working on tutorial problems together; some students even started to travel together. The fact that they had permission to work in paired seemed to generate constructive conversion between the students during practicals, and indeed tutorials, where they were talking to one another to actively work through problem, bouncing ideas of one another and coming up with the optimal solution. This is one of the clear goals of inquiry based learning in that students work together and discuss problems and are therefore learning from each other in a deep and meaningful manner.

In was also found that paired programming suits the female members of the class very well. Traditionally females felt out-numbered by males in computer science and related disciplines. Working in pairs (and they all worked with another female) gave them the confidence to work through problems together and also to ask questions of the academics together; this is vital in the challenge of retaining female students within these domains.

The use of pair programming enabled practicals to be assessed in an easy manner particularly with large module sizes and as the module size increases, it is possible to increase from pairs to small groups. It should be noted however, that although the pair programming did encourage students to attend practical and lecture and the module coursework marks were notably better than previous years, this was not reflected in the examination marks. The examination marks were similar to previous years with no significant improvement noted.

5. DISCUSSION

The idea behind this project was to introduce a practice in the Algorithmic Programming II module that would encourage the students to engage and participate in the module and ultimately improve first year retention. Now we need to consider: did we achieve this? Did we go some way towards achieving this?

Within the U.K. and Ireland, the issue of student retention is prominent. However, throughout the academic year we can often determine which students are likely to fail first year based on their attendance and amount of engagement in the degree program. This is particularly prevalent in computer science related disciplines as students often learn Information and Communication Technologies ICT in schools and think this is computer science. Therefore when entering the degree program many discover it is not what they first thought.

The main reason for introducing pair programming into the programming module was to get students to engage with each other and the module, improving attendance, performance and overall retention. The attendance at practicals improved in comparison to the attendance in AY08/09. As illustrated in Table 1, the practical assessment marks increased on average by almost 54% compared with the expectation. Table 1 also illustrated improved performance among the females in the class which is also an expectation of pair programming.

The next major aim of this project was to enhance learning by encouraging students to attend their practicals and practical assessments and therefore develop a better understanding of programming than in previous years. The improvement in learning would have been determined by improved class test and examination marks. Unfortunately this was not the case. The average class test mark dropped by 2% and the overall examination marks were similar to previous years. However, this was primarily due to approximately 8 students who never engaged in the module and did not sit the exam. This is still a relevant point as the aim of the project was to improve engagement and attendance. Pair programming cannot benefit those who do not take part. However the students who participated in pair programming clearly benefitted.

6. CONCLUSION

We have presented an approach to embedding inquiry based learning into algorithmic programming modules via the use of pair programming. With respect to pair programming students work together in pairs and are also assessed in pairs in their practical assessments. We have presented initial findings that illustrate that the use of pair programming encouraged students to attend lectures and practicals and also, overall, the module assessment marks were significantly higher than expected based on the cohorts experience performance and the assessment marks from previous years. However, it is also evident that the use of pair programming did not improve the examination marks in any significant way.

APPENDIX A

In order to obtain the expected results, the total possible attendance from the first semester's module was obtained by multiplying the number of students by the number of weeks. Next the attendance every week of the semester was summed and divided by the total possible attendance to obtain the attendance percentage. This was carried out for the current and previous academic years. The comparison of these first semester attendance records was then used against the previous year's Algorithmic Programming II module attendance to achieve the expected attendance for the current Algorithmic Programming II module. The same method

was used to obtain expectations of practical assessment results. The benefit of using this method is that a difference is quantified and applied to the same students in different years for attendance, assessments and class tests. This way an amount of improvement or decline in the students without pairing is found, leaving the difference in the analysis as the result of pairing; not ability.

7. ACKNOWLEDGMENTS

The authors would like to thank the Higher Education Academy Information and Computer Science Subject Centre for funding this project.

8. REFERENCES

1. Arisholm, E., Gallis, H., Tore, D., Sjoberg, D.I.K., “Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise” *IEEE Trans on Software Engineering,* Vol. 33, No. 2, Feb 2007.
2. Braught, G., Eby, L.M., Wahls, T., “The Effects of Pair-Programming on Individual Programming Skill” *SIGCSE*, March 2008.
3. Cockburn A., Williams, L., “The Costs and Benefits of Pair Programming” The XP Series, Extreme Programming Examined, pp. 223-243, 2001
4. Dyba, T., Arisholm, E., Sjoberg, D.I.K., Hannay, J.E., Shull, F., “Are Two Heads Better than One? On the Effectiveness of Pair Programming” *IEEE Software*, Vol. 24, No. 6, pp12-15, Dec 2007.
5. McDowell, C., Werner, L., Bullock, H. E., Fernald, J., “Pair Programming Improves Student Retention, Confidence and Program Quality” *Communications of the ACM*, Vol. 49, No. 8, August 2006.
6. Prince, M., Felder, R.W., “The many faces of inductive teaching and learning” Journal of College Science Teaching, Vol. 36, No. 5, pp14-20, March/April 2007
7. Prince, M., Felder, R.W., “Inductive teaching and learning methods: Definitions, comparisons, and research bases”. Journal of Engineering Education 95 (2): 123–38, 2006
8. Rice, P., “Facilitating learning in small groups” http://www.heacademy.ac.uk/
9. Sanders, D., “Student Perceptions of the Suitability of Extreme and Pair Programming” Computer Science Education, 2003
10. Simon, B., Hanks, B., “First-Year Students’ Impressions of Pair Programming in CS1” *ACM Journal on Educational Resources in Computing,* Vol. 4, No. 7, Jan 2008.
11. The Sparks Foundation (2010). *Why promote inquiry based science programming?* Retrieved August 6th 2010. http://scienceexplorers.com/wp-content/uploads/promoting-inquiry-based-learning.pdf
12. Williams, L., "Integrating Pair Programming into a Software Development Process" (PDF). *14th Conference on Software Engineering Education and Training*, 2001