LEARNING TO PROGRAM
From Pear-Shaped to Pairs

Donna Teague, Paul Roe
Queensland University of Technology, Brisbane, Australia
d.teague@qut.edu.au, p.roe@qut.edu.au

Keywords: Pair programming, collaborative learning, learning to program.

Abstract: The consistently high failure rate in Queensland University of Technology's introductory programming subject reflects a similar dilemma facing other universities worldwide. Experiments were conducted to quantify the effectiveness of collaborative learning on introductory level programming students over a number of semesters, replicating previous studies in this area. A selection of workshops in the introductory programming subject required students to problem-solve and program in pairs, mimicking the eXtreme Programming concept of *pair programming*. The failure rate for the subject fell from what had been an average of 30% since 2003 (with a high of 41% in 2006), to just 5% for those students who worked consistently in pairs.

1 INTRODUCTION

Like many universities internationally, in recent years enrolments in Queensland University of Technology's (QUT) Information Technology (IT) degree course have taken a dramatic nose dive, leveling off more recently but with little promise of gaining significant ground in the near future. Attrition from IT courses is historically high (Kinnunen, P., Malmi, L. 2006; Biggers, M., Brauer, A. et al. 2008), particularly for women and other minority groups for whom there is often poor representation to begin with (Cohoon, J.M. 2002; Fisher, A., Margolis, J. 2002; Lewis, S., McKay, J. et al. 2006; Murphy, L., McCauley, R. et al. 2006; Reges, S. 2006; Varma, R. 2006; Vilner, T., Zur, E. 2006).

Commonly offered as a first year core subject, introductory programming subjects have an alarming failure rate (Sheard, J., Hagan, D. 1998; Robins, A., Rountree, J. et al. 2003). The serial nature of programming with sequential dependencies between topics has a bottleneck effect on a student's progression through a subject or course of subjects (eg from CS1 to CS2) if foundation or prerequisite skills are not acquired.

Since 2003 an average of 31% of students were failing QUT's introductory programming subject. Attrition from this Australian university's IT courses was increasing and enrolments poor. These abysmal statistics prompted research into the barriers to first year students learning to program.

This paper documents the results of pair-programming experiments conducted over two semesters at QUT to quantify the effectiveness of collaborative learning on introductory level programming students. A selection of workshops in the introductory programming subject required students to problem-solve and program in pairs, mimicking the eXtreme Programming concept of *pair programming* (Beck, K. 2005).

In the final semester of the experiment, only 5% of the paired students failed the subject, compared to a failure rate of 20% for non-paired students. Students participating in the experiment not only achieved better overall results in the subject, but they also performed better in the subject's final exam.

These results indicate that the paired students were able to independently apply their knowledge to new problems, contrary to the observations in a similar study (McDowell, C., Werner, L. et al. 2002),. However, our results concur with more recent findings that students who pair-programmed were more likely to complete the course successfully (Braught, G., Eby, L.M. et al. 2008).

2 BARRIERS TO LEARNING

Literature indicates that first year students in particular face not only cognitive challenges with complex topics like programming, but also a range of social and cultural issues during their transition into university (Cohoon, J.M. 2002; Fisher, A., Margolis, J. 2002; Lahtinen, E., Ala-Mutka, K. et al. 2005). These barriers are likely to impede the students' full potential being realized or have more significant negative effects on their learning outcome resulting in failure or withdrawal from the unit, or withdrawal from IT degree entirely.

Collaborative learning is known to provide benefits to students including generating enhanced interest in the material, engagement in the learning environment, greater overall achievement and a more enjoyable learning experience (Wilson, J.D., Hoskin, N. et al. 1993; Gokhale, A.A. 1995; Williams, L., Kessler, R.R. 2000; McKinney, D., Denton, L.F. 2006).

Consistently in first year IT subjects at QUT, attendance levels at scheduled lectures, tutorials and workshops dramatically decline through the semester. In the first week of semester 1 2007, the introductory programming subject saw on average 80% of students attending workshops, and by the end of semester the average attendance rate at workshops was only 16%. Subsequent semesters experienced a similar pattern of attendance.

It seems to the authors that introductory programming students (at least at QUT) are reluctant to seriously embrace the advice offered by academic staff for successfully completing the subject. During our experiment, those students who attended scheduled lectures on average spent about half the recommended time per week studying the programming subject. Each week, on average only about half the students attending lectures could say they had studied or practised the material introduced in the previous week's lecture at all. These responses suggest that a 'devil may care' attitude may be responsible for students deferring any significant effort or focus in the course material until the last possible moment. Not unexpectedly, many of them end up struggling to complete complex programming projects in a very limited amount of time. They find themselves with little of the working knowledge required to solve the assessment task. Elevated stress levels compound the problem often resulting in the student's inability to successfully complete the assessment item in time.

Poor grounding in the 'building block' basics of programming like variable declaration, function definition and parameter passing in the early weeks of semester make the more advanced topics of loops, recursion and abstract data types almost impossible to grasp. Even with the 'wake-up' call of a failed first assessment item and a renewed enthusiasm for putting in some real effort, there is all too often little chance to catch up on the workload in time to salvage a decent grade for the subject. The student is in danger of losing confidence in their own ability and disengaging from the subject altogether.

Why do students fail to engage in the first place? One possibility is the stark contrast between the closely monitored high school environment and the adult world of university. Adolescence is characterised by growing dissatisfaction with, and resistance to, authority (White, A.M. 2004), and it is during this stage that students find themselves with the sole responsibility for their learning. This could present the immature student with the opportunity to make poor judgement calls in terms of their commitment to, and organisation and planning of their university obligations (Begley, S. 2000).

But it is not only school-leavers who fail to engage. Those students who don't fit the IT student stereotype include not only women, but mature-age students and others who see studies in IT as complementary to their career aspirations, rather than the focus thereof (Vilner, T., Zur, E. 2006; Peckham, J., Stephenson, P.D. et al. 2007). These students may initially have a better study ethic, but can struggle with a lack of supporting social structure in the learning environment (Cohoon, J.M. 2002) and disinterest in or inability to relate to the learning material (Fisher, A., Margolis, J. 2002).

2.1 Engaging Students

"*I hear and I forget. I see and I remember. I do and I understand.*" [Confucius]

The literature on CS education embraces the notion that lots of hands-on practice and experimentation is especially important for novice programmers (Hassinen, M., Mäyrä, H. 2006), because their knowledge of programming is not passively absorbed through texts and lectures, but rather actively constructed via their own practical experiences (Bruner, J. 1990; Ben-Ari, M. 1998; Huitt, W. 2003).

Collaborative learning establishes an environment conducive to learning and addresses the social and cultural barriers facing first year students and enhances their learning experience (Wilson, J.D., Hoskin, N. et al. 1993; Gokhale, A.A. 1995;

Williams, L., Kessler, R.R. 2000; McDowell, C., Werner, L. et al. 2002; Gehringer, E.F., Deibel, K. et al. 2006). Students benefit from peer support while learning, and at the same time are motivated by peer pressure and a sense of purpose and belonging (McKinney, D., Denton, L.F. 2006).

To further support this literature, first year IT students at QUT were surveyed in 2007 and an overwhelming number responded that they believed learning programming collaboratively would not only have a positive influence on their confidence and ability to develop sound programming skills, but would also make studying programming more engaging and fun (Teague, D., Roe, P. 2008). Hanks (2006) had also reported that the attitude of students to pair programming was mostly positive, and particularly beneficial to women.

Using pair programming in the learning environment has been documented as having significant educational benefits including active learning and improved retention, program quality, and confidence in the solution (McDowell, C., Werner, L. et al. 2002; Williams, L., Wiebe, E. et al. 2002; Nagappan, N., Williams, L. et al. 2003; McDowell, C., Werner, L. et al. 2006; Mendes, E., Al-Fakhri, L. et al. 2006). Students also find programming in pairs creates a social rather than competitive environment which promotes interaction and lends twice as much brain power and an extra set of eyes to a programming exercise (Simon, B., Hanks, B. 2007).

3 GOING PAIR-SHAPED

Following the 2007 survey and aiming to develop a collaborative learning environment to support novice programmers (Werner, L.L., Hanks, B. et al. 2004; Keefe, K., Sheard, J. et al. 2006; Bagley, C.A., Chou, C.C. 2007) an experiment was conducted over two semesters involving introductory level programming students at QUT

The hypothesis tested was that pair-programming style collaborative learning has a positive effect on students' learning outcome.

3.1 The Experimental Environment

The experiments were conducted over two semesters, each of 13 weeks.

ITB001 (*Problem Solving and Programming*) is a core programming subject of QUT's IT Bachelor degree and is perhaps the equivalent of CS1 in the US. This subject is offered by the university every semester, but is normally undertaken by students in the first semester of their degree course. Students enrolling in this subject in the second semester of any year consist mainly of a small number starting their course mid-year and those who initially fail the subject and are forced to repeat it.

ITB001 represented 25% of a full-time study workload, and during the experiment weekly contact consisted of a two hour lecture and a two hour workshop. Workshops involved students completing programming exercises to reinforce in a practical way the material previously introduced in a lecture. Attendance at workshops was strongly encouraged but was neither obligatory nor counted towards final grades. Apart from lectures and workshops, all students were expected (according to university guidelines) to dedicate an extra 8 hours per week to self-directed study for a total of 12 hours study per unit per week.

The assessment for this unit consisted of two or three individual assignments of increasing difficulty (total of 50%) and an end of semester written exam (50%).

Workshops for semester 2, 2007 were conducted without the use of computers, where students concentrated more heavily on the analysis, problem solving and design of their exercise solutions on paper. 2008 saw workshops conducted in laboratories with exercises completed on computers.

The experimental subjects were those students who had previously self-allocated to any one of a number of workshops where the first author was on the teaching staff. These students were instructed on the logistics of pair programming during class and were also encouraged to continue collaboration with their partner outside normal class times.

The control group became those students in other workshops, in which no collaborative learning support was given. It is worth noting that although the first author was tutoring the paired students, items of assessment for grading were distributed to teaching staff on a random basis and therefore that author would have been responsible for grading both students from the paired as well as unpaired workshops over the course of the experiment.

3.2 Pair Selection

Pairs were determined by self-selection. For some students, the prospect of being able to work with a friend throughout semester was something they relished. Others were initially more reluctant to pair because they either had not formed friendships with anyone in the workshop or they simply preferred to work alone. These students were asked to discuss their computing and programming experience (if any), and the teaching assistant then helped them pair with someone of similar skill levels.

Initially all students were paired in the workshops where the experiment was undertaken. However, workshop numbers were large and attendance fluctuated dramatically. As the semester progressed, it proved more difficult to manage the pairs as inevitably one or other of them was away and/or had dropped out. If students resisted the pairing– or their partner deserted them and they expressed a preference to work alone, they were allowed to do so. A small number (9%) of the students, who regularly attended workshops where the experiment was conducted, worked individually and were included in the control group.

A student was considered to be a "paired" if they attended six or more of the weekly workshops during semester (ie approximately half). It is reasonable to assume that these students would have at least been exposed to the pair programming pedagogy, and had experienced studying in a collaborative environment. Results of the experiment may easily have been skewed in favour of pairing, had the subjects been only those paired students who attended *most* of the workshops, as regularly attending workshops could be a contributing factor for success. All other students completing ITB001 during the two semesters of the experiment were considered to be "non-paired".

3.3 Pair Programming

Students in these workshops formed pairs from the first week of semester and were provided with literature concerning the benefits of collaborative learning. They were also given verbal and written instructions on pair-programming together with background information to read. Teaching assistants instructed on the logistics of collaborating with their partner according to the eXtreme Programming concept of pair programming (Williams, L., Kessler, R. 2003). Each student in the pair assumed a different role for each exercise (or in the case of larger exercises, the roles were swapped at intervals of 15 minutes or so):

– the **"driver"** took control of the keyboard/pen: eg recording the algorithm; writing code; debugging and executing the code
– the **"observer"** was responsible for thinking strategically, asking questions, watching for errors, suggesting alternatives, and providing technical input

Each week these students were reminded of the distinct roles each partner in the pair was to play. Teaching assistants directed the students at regular intervals to swap roles and encouraged intensive and continuous interaction between the paired students.

The pairing experiments were formally conducted during the two hour weekly workshop and continued for the duration of each semester. Students were encouraged to continue their paired collaboration outside the workshops by completing unfinished workshop exercises and work on the analysis and problem-solving of their assignments.

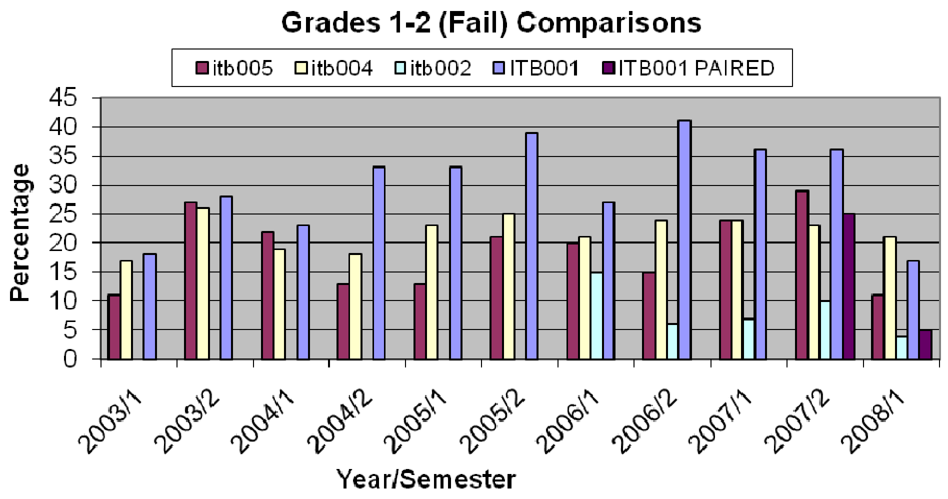

Figure 1: QUT Student Failure Rates - First Year Subjects.

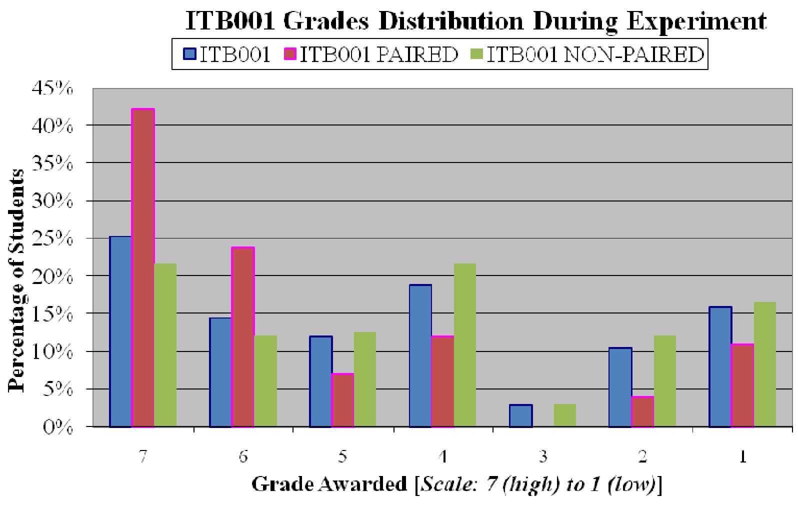

Figure 2: Distribution of grades.

As all assignments were for individual submission, collaboration between students was forbidden past the design stage.

Table 1. Pairing experiment student numbers.

Semester	ITB001 Workshops		Number of Students	
	Paired	Non-Paired	Paired	Non-Paired
2, 2007	2	4	16	77
1, 2008	4	14	64	274

4 RESULTS

4.1 Grades Awarded

Figure 1 plots student grades awarded at QUT for the first four core subjects of its Bachelor of IT degree from 2003. ITB005, ITB004 and ITB002 are subjects normally undertaken concurrently with ITB001. The ITB001 data shown in this figure represents the entire cohort of ITB001 students (paired and non-paired), while ITB001 PAIRED show the results for paired students only.

Prior to the pairing experiment, ITB001's failure rate averaged 30%, with a peak in 2006 of 41%.

Amongst the paired student population, there was a dramatic fall in failure rate for ITB001 in both semesters of the experiment, dropping to just 5% in semester 1, 2008 ($n = 431$, $p < .001$). At the other end of the spectrum, 70% of paired students achieved a grade of 6 or 7 on a scale of 1 (low) to 7 (high).

Figure 2 summarises the distribution of grades awarded for the entire cohort of ITB001 students, paired and non-paired ITB001 students during the experiment.

4.2 Exam Results

Paired students not only achieved better overall grades than the non-paired students, but they significantly outperformed the control groups in all sections of the final exam which included comprehension, tracing, problem solving and code writing questions ($n = 431$, $p < .001$).

4.3 Predicting Results without Pairs

In order to estimate what grades the paired students may have achieved had they *not* participated in the pairing experiment, a comparison is made between ITB001 and another subject of a comparable level technical nature, ITB004 *Database Systems*. ITB004 teaches database design, the concepts and terminology relating to databases, and involves writing data manipulation statements in Structured Query Language (SQL). Each week, ITB004 conducted a two hour lecture and two hours of workshops. Although small group discussion was encouraged during one hour of the workshops, no formal collaborative learning structure was in place for these students. Assessment for ITB004 consists of individual assignments (total 35%) and a final end of semester exam (55%), with a further 10% awarded for workshop participation.

Final results for students who completed these two subjects consecutively during the experiment period (whether they paired or not) were compared. Therefore, it is reasonable to assume that that study of each of these subjects was influenced to a similar degree for example by family and social commitments, employment, competing study commitments as well as attitude to and motivation for study.

Figure 3 shows the relationship between students' grades for both subjects, by graphing the variation in grade between ITB001 and ITB004. There were a similar number of unpaired students who achieved a higher grade in ITB004 as those who performed better in ITB001. This is evidenced by the symmetry of the grades curve for that subject.

Of those 105 unpaired students, 42% achieved a similar result in both subjects, and were awarded the same grade for both. 28.5% performed better in ITB004 and 29.5% performed better in ITB001.

By comparison, a greater proportion of students who took part in the pairing experiment (the paired students) achieved a better grade in ITB001 than in ITB004. Although a significant number (52%) attained the same grade for both subjects, more than 38% of students performed better in ITB001 while just under 10% performed better in ITB004.

This comparison of student grades for two similarly technical subjects further supports the theory that learning programming in a collaborative environment involving pair-programming had a positive effect on student results. One might also expect that students who enjoyed the benefits of pair-programming in ITB001 may well have employed those collaborative learning skills to their ITB004 studies and had a positive effect on their grade for ITB004 too. Had the experiment been able to eliminate any copy-cat effect in ITB004, the results shown in the comparison of these two subjects may well have been even more convincing.

5 CONCLUSION

The failure rate of students in the introductory programming subject involved in this experiment enjoyed a dramatic fall from a high of 41% to just 5%. Although it is acknowledged that other factors may have contributed to this improvement including teaching staff, subject content, programming language and student cohort, paired students performed significantly better than those who were not paired in the same semester, with exposure to the same subject structure.

Furthermore, given results data from another subject undertaken concurrently by the same students, it is reasonable to suggest that the paired students achieved greater than expected had they not had the support of the pair-programming learning environment.

Students exposed to pair-programming and supported by a collaborative learning environment outperformed the control group of students who worked independently throughout semester in the final exam as well as overall subject results.

6 OBSERVATIONS

"Engaging" in the pair programming experiment involved the students firstly selecting, and then establishing a rapport with another student. Where there existed no significant conflict or imbalance in terms of language, work ethic or skills level, successful social engagement between students had a positive follow-through effect on the business end of the programming tasks each week.

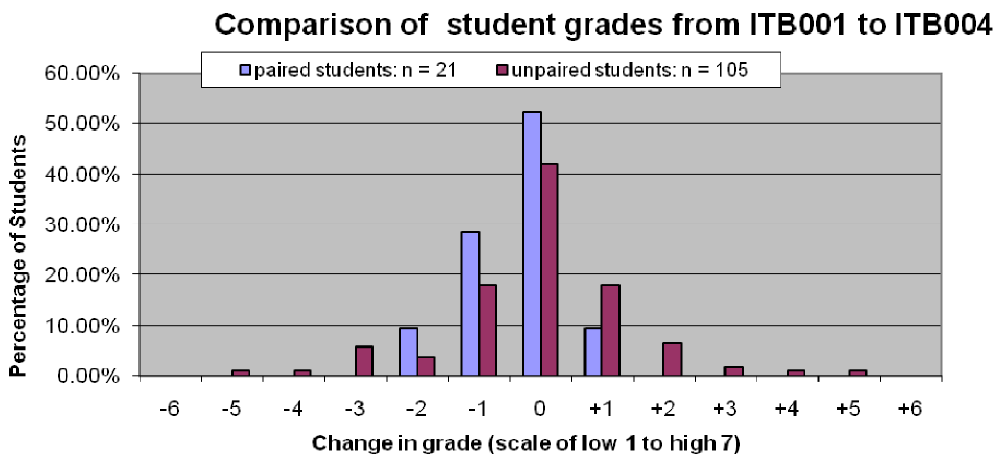

Figure 3: Difference in grades for two units for paired and non-paired students during experiment period.

By virtue of their social interaction, the paired students established a productive learning environment for each other on their level. The ego-charged stereo-typical student was given the opportunity to flex his IT muscles for a peer who may speak the same lingo and appreciate the display of competitive prowess. Alternatively, the student who may have harboured reservations about their ability was able to develop a non-threatening learning environment by pairing with a peer of similar experience and level of confidence in the course material.

Once relationships were formed between the pair, the students unwittingly tended to maintain a two-way support structure by having a more personal reason to attend the workshop and engage in the material: a sense of obligation to their partner. They were provided with not only an *opportunity* to discuss the work and contribute to the pair's progress but there was also an *expectation* by their partner to do so. This peer pressure seems to have more of an influence on the motivation of the novice student than any amount of pressure from the teaching staff. The students' obligation to, and stake in their partner's learning experience had at least as high a priority as any sense of obligation to their own learning outcome. Because it is difficult to play a very passive role in a pair (as opposed to a larger group) students seemed to develop a commendable study ethic while paired.

Collaborative learning generally worked so effectively that it seemed unfortunate that students were not given the opportunity to continue pair-programming throughout development of their assignments. The requirement that 'group assignments' not be incorporated in the subjects' assessment on the basis that they may not accurately reflect an individual's level of acquired skill and contribution may be misplaced. The better performance in the final exam shows that paired students *did* acquire the necessary problem solving and programming skills. Incorporating peer evaluation into a paired assignment could exploit the sense of obligation that developed in well-formed pairs to ensure students contributed adequately, while oral presentation or written examination of the assignment could further ensure that marks are awarded fairly.

7 FURTHER WORK

Further pair-programming experiments over a longer time period would be useful to further support the theory that collaborative learning has a positive effect on student outcome.

In future work, analysis of workshop attendance rates may be useful in order to determine any correlation between such attendance and student outcome.

REFERENCES

Bagley, C.A., Chou, C.C., 2007. Collaboration and the Importance for Novices in Learning java Computer Programming. ITiCSE Conference '07. Dundee, Scotland.

Beck, K., 2005. Extreme programming explained : embrace change Boston, MA, Addison-Wesley.

Begley, S., 2000. Getting Inside a Teen Brain. Newsweek. 135: 58-59.

Ben-Ari, M., 1998. Constructivism in Computer Science Education. Twenty-ninth SIGCSE technical symposium on Computer science education 30(1).

Biggers, M., Brauer, A., Yilmaz, T., 2008. Student Perceptions of Computer Science: A Retention Study Comparing Graduating Seniors vs. CS Leavers. 39th SIGCSE technical symposium on Computer science education, Portland, OR, USA, ACM.

Braught, G., Eby, L.M., Wahls, T., 2008. The Effects of Pair-Programming on Individual Programming Skill. 39th SIGCSE technical symposium on Computer science education (SIGCSE '08), Portland, OR, USA, ACM.

Bruner, J. 1990. Constructivist Theory. Retrieved 19 July, 2007, from http://tip.psychology.org/bruner.html.

Cohoon, J.M., 2002. Recruiting and Retaining Women in Undergraduate Computing Majors. ACM SIGCSE Bulletin 34(2).

Cohoon, J.M., 2002. Women in CS and Biology. ACM SIGCSE Bulletin, Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education SIGCSE '02 34(1).

Fisher, A., Margolis, J., 2002. Unlocking the clubhouse: the Carnegie Mellon experience ACM SIGCSE Bulletin 34(2).

Gehringer, E.F., Deibel, K., Whittington, K.J., 2006. Panel: Cooperative Learning—Beyond Pair Programming and Team Projects. SIGCSE 2006 Technical Symposium on Computer Science Education. Houston, Texas USA.

Gokhale, A.A., 1995. Collaborative Learning Enhances Critical Thinking. Journal of Technology Education 7(1): 22-30.

Hanks, B., 2006. Student Attitudes toward Pair Programming. ITiCSE 06: Proceedings of the 11th annual conference on Innovation and technology in computer science education.

Hassinen, M., Mäyrä, H., 2006. Learning Programming by Programming. 6th Baltic Sea Conference on Computing Education Research, Koli Calling.

Huitt, W. 2003. Constructivism. Educational Psychology Interactive. Retrieved 19 July 2007, 2007, from http://chiron.valdosta.edu/whuitt/col/cogsys/construct.html.

Keefe, K., Sheard, J., Dick, M., 2006. Adopting XP practices for teaching object oriented programming. ACM International Conference, Hobart, Australia, ACM.

Kinnunen, P., Malmi, L., 2006. Why Students Drop Out CS1 Course? 2006 international workshop on Computing education research ICER '06.

Lahtinen, E., Ala-Mutka, K., Järvinen, H.-M., 2005. A Study of the Difficulties of Novice Programmers. 10th annual SIGCSE conference on Innovation and technology in computer science education ITiCSE '05.

Lewis, S., McKay, J., Lang, C., 2006. The Next Wave of Gender Projects in IT Curriculum and Teaching at Universities. Eighth Australasian Computer Education Conference (ACE2006), Hobart, Tasmania, Australia, ACS.

McDowell, C., Werner, L., Bullock, H., Fernald, J., 2002. The Effects of Pair-Programming on Performance in an Introductory Programming Course. 33rd SIGCSE technical symposium on Computer science education. Cincinnati, Kentucky ACM.

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

McKinney, D., Denton, L.F., 2006. Developing Collaborative Skills Early in the CS Curriculum in a Laboratory Environment. SIGCSE 2006 Technical Symposium on Computer Science Education. Houston, Texas, USA.

Mendes, E., Al-Fakhri, L., Luxton-Reilly, A., 2006. A Replicated Experiment of Pair-Programming in a 2nd-year Software Development and Design Computer Science Course. ITiCSE 06: Proceedings of the 11th annual conference on Innovation and technology in computer science education Bologna, Italy.

Murphy, L., McCauley, R., Westbrook, S., 2006. Women Catch Up: Gender Differences in Learning Programming Concepts. SIGCSE 2006 Technical Symposium on Computer Science Education. Houston, Texas USA.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., Balik, S., 2003. Improving the CS1 Experience with Pair Programming. 34th SIGCSE technical symposium on Computer science

Peckham, J., Stephenson, P.D., Harlow, L.L., Stuart, D.A., Silver, B., Mederer, H., 2007. Broadening participation in computing: issues and challenges. ACM SIGCSE Bulletin, Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education ITiCSE '07 39(3).

Reges, S., 2006. Base to basics in CS1 and CS2. SIGCSE'06, Houston, Texas, USA, ACM.

Robins, A., Rountree, J., Rountree, N., 2003. Learning and Teaching Programming: A Review and Discussion. Journal of Computer Science Education 13(2): 137-172.

Sheard, J., Hagan, D., 1998. Our failing students: a study of a repeat group. Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education: Changing the delivery of computer science education ITiCSE '98.

Simon, B., Hanks, B., 2007. First Year Students' Impressions of Pair Programming in CS1. Third International Computing Education Research Workshop. Georgia Institute of Technology, Atlanta, GA USA, ACM.

Teague, D., Roe, P., 2008. Collaborative learning: towards a solution for novice programmers. Proceedings of the tenth conference on Australasian computing education. Wollongong, NSW, Australia, ACS.

Varma, R., 2006. Making Computer Science Minority-Friendly. Communications of the ACM 49(2).

Vilner, T., Zur, E., 2006. Once She Makes it, She is There: Gender Differences in Computer Science Study. ITiCSE 06: Proceedings of the 11th annual conference on Innovation and technology in computer science education, Bologna, Italy.

Werner, L.L., Hanks, B., McDowell, C., 2004. Pair Programming Helps Female Computer Science Students. Journal on Educational Resources in Computing (JERIC) 4(1).

White, A.M. 2004. Adolescence: What, why and when? Retrieved 26-11-2008, 2008, from http://www.duke.edu/~amwhite/Adolescence/adolescent2.html.

Williams, L., Kessler, R., 2003. Pair Programming Illuminated. Boston, Addison-Wesley.

Williams, L., Kessler, R.R., 2000. The Effects of "Pair-Pressure" and "Pair-Learning" on Software Engineering Education. Proceedings of 13th Conference on Software Engineering Education & Training, 2000.

Williams, L., Wiebe, E., Yang, K., Ferzli, M., Miller, C., 2002. In Support of Pair Programming in the Introductory Computer Science Course. Computer Science Education 12(3): 197-212.

Wilson, J.D., Hoskin, N., Nosek, J.T., 1993. The Benefits of Collaboration for Student Programmers. 24th SIGCSE Technical Symposium on Computer Science Education SIGCSE 1993, Indianapolis, Indiana US, ACM Press.