# Enhancing Computer Learning Activities Using Pair-Programming Techniques

**Conference Paper** · December 2014

DOI: 10.13140/2.1.5018.7844



**2 authors**, including:

Mamoun Nawahdah
Birzeit University
**22** PUBLICATIONS **13** CITATIONS

SEE PROFILE

# Enhancing Computer Learning Activities Using Pair-Programming Techniques

Dima Taji
Computer Science Department
Birzeit University
dtaji@birzeit.edu

Mamoun Nawahdah
Faculty of Engineering and Technology
Birzeit University
mnawahdah@birzeit.edu

**Abstract**

Transitioning computer science students from the university environment, where everything had to be done individually, into an environment that would require them to work in pairs might be difficult to most students. In order to make this transition a smoother one, students need to be familiarized with the concept of working together in pairs before they are introduced to the workforce. This motivated the introduction of pair programming as a teaching technique in our computer science labs. Pair programming was attempted in several universities worldwide with various degrees of success. In this paper, we investigate the benefits of practicing pair programming in education and in the industry. We also review some conducted experiments where pair programming was used as a teaching technique. We include the successes and impediments faced in these experiments. This paper revealed that pair programming as a teaching technique might have the ability to improve the quality and level of university students, and make their transition to the industry a smoother one.

## 1 Introduction

Programming has always been considered a solitary activity[6]. This was the way it was taught, and the way it was practiced until 1999, when Kent Beck created extreme programming, and listed pair programming as one of its twelve practices[1]. Nevertheless, solo programming was associated with the waterfall development cycle in software engineering, suggested in 1970, in which a development team would meet with the customer once to get all the requirements of the system, and then design it, and implement its design[5].

The shortcomings of the waterfall development model, including the tendency of customers to constantly change their minds[1], the discovery of mistakes in the testing and debugging phase of the project, which comes later in

the project[6], and limiting each developers knowledge to their own piece of the system[6, 16]. This prompted the search for alternative models such as the agile model[2].

The agile model aims to be flexible and collaborative[3], which includes incorporating many techniques and practices, of which pair programming is standing out for being somewhat controversial. Pair programming may be defined as "*the practice whereby two programmers work together at one computer, collaborating on the same algorithm, code, or test*"[4]. Not only do the programmers work at one computer, but all production code is written with two people looking at one machine, with one keyboard[2].

Each of the two programmers working in pair programming have a designated role. One of the programmers is called the driver. The driver's role is to be the ones who is actively typing at the computer[4], therefore in control of the keyboard. The second role is called the navigator, who reviews the code being typed to catch errors, or make suggestions[4]. The two programmers should continuously switch roles, where the driver becomes the navigator, and the navigator becomes the driver. This allows both members of the pair to benefit from experiencing both roles[18]. In addition, programmers are often encouraged to switch partners between tasks.

Pair programming as a teaching technique may have many advantages. The following section reviews the advantages of pair programming, from previous literature, both in the industry and in education. Next, it gives an overview of how pair programming can be adapted to be used as a teaching technique in programming labs. Finally, it presents the results of some of the conducted experiments and study cases where pair programming was applied as a teaching technique.

## 2 Merits of Pair Programming

Pair programming has many advantages that have been repeated throughout the literature. One of these advantages are improving design quality[4, 6, 7, 13, 14, 16], since two programmers collaborate together on the same piece of code, resulting in improving the quality of the design. Another advantage is reducing defects[6, 7, 16]. This is the result of continuous revision of code as it is being written, which makes it simpler to catch bugs and errors before the development phase is done.

Another advantage of using pair programming technique is improving team communication[6, 16]. Since programming is usually viewed as a solo activity, there is not much communication going on between team members. Nevertheless, since pair programming involves having two programmers sitting at the same machine working on the same piece of code, they will have to communicate with each other. This has an effect of improving the social skills of team members and making their communication together more efficient.

An effect that has been noticed as well is that the code becomes simpler and easier to extend[6]. When two programmers collaborate on writing the

same piece of code, obscure segments will be discussed and improved as a result of this collaboration. In addition, the resulting code will be more extendable than code that only one person writes for the same reason of having more than one person writing it. In addition, some researchers have found that people working using pair programming tend to spend more effort on the tasks they undertake[8], which can be attributed to having another developer continuously reading the code and revising it.

When used as a teaching technique, pair programming has several advantages in addition to those observed in the industry. Students' performance was shown to have improved when using pair programming[12, 14, 15], even when students were required to program individually for their final exam[10, 17]. Working in pairs promotes peer tutoring. Research shows that student may be more receptive to information when coming from their peers as opposed to an instructor[9].

In addition, students produced better quality programs than when working on their own[10, 12, 14, 15]. Students exchange allows them to learn from each others experiences[12], and brainstorm before working[18], therefore allowing them to come up with the most efficient techniques to solve a given problem.

An advantages that has been noticed when surveying students who studied in labs applying pair programming technique is course enjoyment[10, 12, 14, 15]. Students reported that they enjoy working in pairs more than they do on their own. This might be due to the social aspect of pair programming, or even due to the fact that they are producing good code and solving problems with a little less effort. On the same note, pair programming allows for less frustration when programming [18], since it is less likely to get stuck at a problem for long.

Studies have also show that applying pair programming in courses help improve course completion rate[10, 12, 14, 15], where less students were found to drop the course in classes where pair programming is being practiced. This comes as a logical consequence to having a course where they are enjoying the work they are doing, while achieving good results.

# 3 Pair Programming as a Teaching Technique

Research shows that when students take a programming course, they are usually in their first year of university, when it is important to get them used to help them gain the skills needed during their university years as well as integrating into the industry[18]. This is why most experiments carried out with pair programming as a teaching technique is applied to introductory level courses[9, 10, 12, 18]. However, a number of researchers attempted to carry out the experiments on second-year and even advance students as well[11, 13].

This section begins by explaining how the pairs are formed in the computer science labs that use pair programming. Next, it shows how these labs are carried out, and what types of activities take place. Finally, it goes over how students in such labs are usually assessed for their final grades in terms of assignments and exams.

### 3.1 Pair Formation

The way students are paired could affect the dynamics of the pair, and might result in an an effect on the outcome of the experiments. The methods followed for the formation of pairs differed among experiments. They varied between forming pairs according to their level of programming experience[9, 13, 18], random formation[9, 11], letting students form their own pairs[9, 18], or a combination of having students select a number of potential partners, and then assigning one of them according to experience[10].

Teague explains that when students chose the partner with which, they often realized that this was not the partner they actually wanted to work with, or were most compatible with[18]. This is why most researchers prefer having students paired according to their level of programming experience. This is usually decided according to the students' performance and grades in previous related programming courses, or assessing their programming experience[9, 18]. Nevertheless, it can be noted that other researchers claim that the experiments succeed most when students are paired randomly[11].

### 3.2 Pair Programming Model

One of the most critical things when starting a pair programming experiment is identifying the difference between pair programming and team work, in the sense that pair programming does not mean each member of the team being responsible for completing a piece of the project individually and then combining pieces together[18].

In addition, pairs were instructed to constantly switch roles between driver and navigator[9, 12, 18]. The driver's task was to control the mouse and keyboard, as well as compiling and running the code. On the other hand, the navigator constantly looked for syntax and logical errors, and searched for resources and alternatives to the implementation. It was observed that in the later parts of the experiments this switch was taking place more frequently and smoothly than at the beginning[18].

### 3.3 Student Assessment

To evaluate their experiments, researchers took into consideration several aspects. In all the reviewed experiments, students were required to finish a number of assignments for which they were graded[9, 10, 11, 13, 18]. In addition, at least a final exam was given to assess students' performance in the course[10, 11].

A challenge that faced instructors when assessing assignments submitted by a pair was how to distribute the weight among both students submitting the assignment. This was addressed by Khan et al. in [9]. They claimed that interviewing the students individually was time consuming, and thus not favoured. Therefore, they adopted an approach where they would assess the submitted project giving both students the same grade, but incorporating a peer-review

element, where each partner had to evaluate their partner's cooperation and performance on a scale of one to five, which factored in their final grade.

### 3.4 Conclusion

Training students at working in pairs helps preparing them to integrate better in the programming work environment. Previous research shows that applying teaching techniques based on pair programming helps students in getting a feel of what working in the industry is like. In addition, research shows that this technique is beneficial as it increases students' enjoyment of programming courses. It also helps them achieve better grades and improves course completion rates. Finally, it allows them to write programs of a higher quality.

## References


[1] Kent Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, 1999.

[2] Kent Beck. *Extreme programming explained: embrace change*. Addison-Wesley Professional, 2000.

[3] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001.

[4] Andrew Begel and Nachiappan Nagappan. Pair programming: what's in it for me? In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 120–128. ACM, 2008.

[5] Bernd Bruegge and Allen H Dutoit. *Object-Oriented Software Engineering Using UML, Patterns and Java-(Required)*. Prentice Hall, 2004.

[6] Alistair Cockburn and Laurie Williams. The costs and benefits of pair programming. *Extreme programming examined*, pages 223–247, 2000.

[7] Enrico di Bella, Ilenia Fronza, Nattakarn Phaphoom, Alberto Sillitti, Giancarlo Succi, and Jelena Vlasenko. Pair programming and software defects–a large, industrial case study. *Software Engineering, IEEE Transactions on*, 39(7):930–953, 2013.

[8] Omar S Gómez, José L Batún, and Raúl A Aguilar. Pair versus solo programming–an experience report from a course on design of experiments in software engineering. *arXiv preprint arXiv:1306.4245*, 2013.

[9] Shamim Khan, Lydia Ray, Aurelia Smith, and Angkul Kongmunvattana. A pair programming trial in the cs1 lab. In *Proc. Annual International Conference on Computer Science Education: Innovation and Technology (CSEIT)*, pages 6–7, 2010.

[10] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. The effects of pair-programming on performance in an introductory programming course. In *ACM SIGCSE Bulletin*, volume 34, pages 38–42. ACM, 2002.

[11] Emilia Mendes, Lubna Al-Fakhri, and Andrew Luxton-Reilly. A replicated experiment of pair-programming in a 2nd-year software development and design computer science course. In *ACM SIGCSE Bulletin*, volume 38, pages 108–112. ACM, 2006.

[12] Leo Porter, Mark Guzdial, Charlie McDowell, and Beth Simon. Success in introductory programming: what works? *Communications of the ACM*, 56(8):34–36, 2013.

[13] Annu B Prabhakar. Applying pair programming for advanced java course: a different approach. In *Proceedings of the 2011 conference on Information technology education*, pages 319–320. ACM, 2011.

[14] Norsaremah Salleh, Emilia Mendes, and John Grundy. Empirical studies of pair programming for cs/se teaching in higher education: A systematic literature review. *Software Engineering, IEEE Transactions on*, 37(4):509–525, 2011.

[15] Norsaremah Salleh, Emilia Mendes, John Grundy, and Giles St J Burch. The effects of neuroticism on pair programming: an empirical study in the higher education context. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, page 22. ACM, 2010.

[16] Alberto Sillitti, Giancarlo Succi, and Jelena Vlasenko. Understanding the impact of pair programming on developers attention. In *Proceedings of the ICSE2012 Conference, Zurich, Switzerland*, 2012.

[17] Beth Simon and Brian Hanks. First-year students' impressions of pair programming in cs1. *Journal on Educational Resources in Computing (JERIC)*, 7(4):5, 2008.

[18] Madonna Margaret Teague. Pedagogy of introductory computer programming: a people-first approach. 2011.