Pair Programming
Fall 2019

Contents

1   What is Pair Programming?

Simply put, pair programming is "two people working together at a single computer" [4]. The practice has been popularized by a software development methodology called Extreme Programming (XP), and a number of researchers have studied the effects of incorporating pair programming into introductory and higher-level computer courses.

2   Why Pair Program?

Again, simply put:

- You will produce better code [3, 1]

- You will learn more, sharing your ideas with your peers and benefiting from their insights [2]

- You will become better at articulating your thoughts [3]

- You will enjoy your work more and spend less time frustrated [3, 1]

- You will be better prepared for more complicated software engineering tasks, both in school and beyond, where collaboration is imperative to success [2]

3   How to Pair Program

We first introduce the practice of pair programming, and then follow up with an informal description of the spirit behind it with the hope that a thorough understanding of its goals will help you and a partner work together productively and harmoniously. A successful pair-programmer not only knows and adheres to the formal guidelines, but also is able to resolve intra-partner stress through constructive communication.

3.1   The Formal Description

When two programmers work with with one keyboard, a division of labor is necessary. Originally, XP had one programmer type on the left half of the keyboard and the other on the right. However, that practice was abandoned due to the uncomfortable seating arrangements required and a statistically significant increase in repetitive stress injury (RSI). A perhaps more natural division of roles developed in its place:

The Driver Responsible for typing, moving the mouse, etc.

The Navigator[1] Responsible for reviewing the driver's work. In addition to catching incidental mistakes (that are nonetheless tedious to track down when solo-programming), the navigator considers the code at a more strategic level: how will this fit with the rest of the code? Will this implementation require changes elsewhere? Could we design this program better?

Every fifteen minutes or so, the pair switches roles by sliding the keyboard over [1]. That's a pretty loaded sentence, so let's break it down:

Every fifteen minutes or so You can wait for a natural breaking point [3], but no one should drive more than 20 minutes or less than 10.

the pair switches roles To avoid animosity, ensure the integrity of the team, and keep pair programming fun, it's important that both partners spend an equal amount of time in each role. Some programmers may enjoy driving better, others navigating; however, neither partner does the other a favor by letting them spend more time in a fixed role.

by sliding the keyboard Noticeably, not by rearranging the chairs and adjusting the monitor. That is to say, regardless of roles, you are both working together on the problem at hand. As such you are seated next to each other with the monitor adjusted so you can both see the screen easily. All that's required when you switch, then, is to push the keyboard a few inches this way or that. [1]

It is important to distinguish pair programming from another distinct cooperative division of labor: divide-and-conquer. In this latter strategy, two people responsible for completing a single task break down the task into smaller pieces, partition the resulting subtasks among themselves, and each works on "their part" separately. This strategy can be used successfully, but it should not be confused with pair programming.

3.2   The Informal Description

Here, we outline various practices and perspectives that provide a foundation for your pair programming experience. Knowing that you and your partner have both read this, the two of you will have a common expectation about the pair programming experience and should be able to work together more effectively. These practices and perspectives are inspired principally by a well-written essay that programmers at every level are encouraged to read [1] .

3.2.1   Respect each other's time

Show up *on time*. Plan ahead—eat *before* you arrive. Don't text or check email during your pair programming sessions; stay focused on the joint task at hand.

Respect involves communication. If your partner is often late, bring it up. If you receive a call that you must answer, let your partner know and reschedule your meeting, acknowledging that you've inconvenienced your partner.

3.2.2   Be a team player

Take collective ownership of the code you and your partner are writing, abandoning the notion of "my part" and "your part." And, in light of that view, make sure you speak up when you think an error's been introduced, and don't be too proud to admit a mistake. And finally, if you work on any code alone (for example, if a solution comes to you in a dream, or in the shower), review it together line-by-line before incorporating it into your program. Often it's useful for the two of you to rewrite solo code from scratch (you'd be surprised how many errors you can catch that way).

Again, communication is crucial. Perhaps you're having a hard time adjusting to pair programming, or your partner continues to say "my" instead of "our" work. Speak your mind and work through any problems. Offer to cease driving when it's time ("Hey, would you like a go at the keyboard?"), and remind your partner it's time to switch when you're navigator ("Mind if I drive for a while?").

3.2.3   Be open-minded and light-hearted

One of the most important predictors of success in pair programming is buy-in: if you are determined to make the practice fail, it will. Choose a healthy perspective: laugh at your mistakes, apologize if you hurt your partner's feelings, and, more generally, look at pair programming as an opportunity to learn. Also, pair programming can be a pretty demanding activity, so take breaks when you need them. Perhaps a five-minute microbreak is all you need: to check email, get a drink of water, or catch some fresh air suffices. Or maybe you would like to take an hour off to get dinner.

3.2.4   Seek advice when you need it

It's important to realize that you and your partner are not alone: if the two of you are having trouble working together, it's important to let an HTA or the professor know so she or he can provide further guidance. The HTAs and professors are familiar with the pair programming philosophy and the difficulties that may arise. They will be able to assist in conflict resolution and/or help you and your partner re-pair if necessary.

It is not acceptable in pair programming for a single person to do all (or even most) of the work and then add his partner's name. Academic honesty is always more important than fulfilling a pair programming requirement. If your partner is unwilling to help or fails to show up at scheduled meetings, or if your partner is unwilling or resistant to letting you contribute your share of the work, contact an HTA or the professor. Asking for help is not an inflammatory action, but an important step in conflict resolution.

4   Pair-Programming in CS 17/ 18

> Given enough eyeballs, all bugs are shallow.
> —Eric Raymond

To summarize the collaboration policy included in the course missive:

- On quizzes, you can talk to your neighbors but must write up your own solution.

- In labs, you are required to pair program. Follow the methodology outlined in this document. Note that you are also required to program with a different partner each lab.

- On homeworks, you may not pair program. However, you are encouraged to talk about ideas with classmates; just don't take away notes from such brainstorming sessions.

- For projects, you must work together as a pair. Although you may not be coding, you are expected to follow the methodology outlined above; you may simply be sharing a notebook or a whiteboard instead of a screen.

- On exams, no collaboration of any sort is permitted.

Therefore, your pair programming experiences will chiefly be in labs and on projects.

References

[1] Laurie A. Williams and Robert R. Kessler. *All I Really Need to Know About Pair Programming I Learned in Kindergarten.* vol. 32, no. 5, Communications of the ACM, 2000, pp. 108-114.

[2] Jennifer Bevan and Linda Werner and Charlie McDowell. *Guidelines for the Use of Pair Programming in a Freshman Programming Class. Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET'02)*, Covington: IEEE, 2002.

[3] Laurie Williams and Eric Wiebe and Kai Yang and Miriam Ferzli and Carol Miller. "In Support of Pair Programming in the Introductory Computer Science Course." *Computer Science Education*, vol. 12, no. 3, 2002, pp. 192-212.

[4] Extreme Programming: A Gentle Introduction, `http://www.extremeprogramming.org/` 2007.

Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS 17 document by filling out the anonymous feedback form: `http://cs.brown.edu/courses/csci0170/feedback`.