**CSC 226 How to Pair Program**

This pair programming summary is derived from "All I Really Need to Know about Pair Programming I Learned in Kindergarten" by Laurie A. Williams and Robert R. Kessler. You are welcome to read the full paper. It starts with lots of data supporting the use of pair programming, some of it with computer jargon you may not want to worry about right now, but it elaborates on many of the points more than this summary.

**Share everything.** All contributions, whether leading to great results or errors, are "ours", not "yours" or "mine".

**Play fair.** One person "drives" (has control of the keyboard or is recording design ideas) while the other "navigates" by continuously reviewing the work and planning ahead. Even when one programmer is significantly more experienced than the other, it is important to take turns "driving," lest the observer become disjoint, feel out of the loop or unimportant. The person not driving is not a passive observer, instead is always active and engaged, checking over the work and thinking ahead.

**Don't hit your partner.** But make sure your partner stays focused and on-task.

**Put things back where they belong.** Put negative judgments in the trash: Be positive about you and your partner. For both, this is an opportunity to improve.

**Clean up your mess.** Watch over the shoulder and you are really likely to find a large number of your errors!

**Don't take things too seriously.** If your partner picks out a bunch of errors as you type, be glad there are two of you. But do not always agree. Have healthy disagreement/debate. Finding the fine balance takes adjustment.

**Say you're sorry when you hurt somebody while moving furniture.** Sit side-by-side and program, simultaneously viewing the computer screen and swap the keyboard and mouse back and forth. Slide the keyboard -- don't move the chairs.

**Wash your hands of skepticism before you start.** Being skeptical can be a self-fulfilling prophesy. Buying in and "jelling" as a team can lead to a whole that is greater than the sum of the parts.

**Flush.** If you work on some parts independently either discard them (flush) and start over together or have the partner very carefully review the work with you.

**Warm cookies and cold milk are good for you.** Periodically, taking a break is important for maintaining the stamina for another round of productive pair programming.

**Be aware of the power of two brains.** Experience show that, together, a pair will come up with more than twice as many possible solutions than the two would have working alone. They will then proceed to more quickly narrow in on the "best" solution and will implement it more quickly and with better quality.

---

Modified from *How to Pair Program* by Dr. Andrew Harrington: http://anh.cs.luc.edu/150/Kindergarten.html

**In case you can't tell…. This article is *sarcasm*.**

## 10 Ways to Kill Pair Programming

by Byron Sommardahl published on Friday August 27, 2010

If you have a hard time working with other humans or find that other humans have a hard time working with you, you may want to continue reading. If you value your own opinion over any other regardless of validity, then listen up. If you are a better coder than anyone on your team and you want to keep it that way, then lean in closer. There is a software development technique that is gaining popularity amongst the "agile crazies" that threatens your quiet, peaceful, lonely existence. They call this technique, "Pair Programming."

Wikipedia.com describes "Pair Programming" this way: "Pair programming is an agile software development technique in which two programmers work together at one work station. One types in code while the other reviews each line of code as it is typed in. The person typing is called the driver. The person reviewing the code is called the navigator. The two programmers switch roles frequently."

Sounds nice, right? Well, if I described you correctly in the paragraph above, then the answer is, "wrong!" Working so closely with another programmer is likely to end badly for you. However, "Pair Programming" has been building momentum over the last few years, especially since there has been an increased focus in software engineering practices, code quality, and ethical programming. But who needs any of those things? You make great software and you don't feel like you should have to change the way you work… ever.

It is for those reasons and more that I decided to release this list of useful weapons in the fight against progress.

1. **EXPRESS YOUR SUPERIORITY:** You're a better coder and you know it. Be sure your pairing partner knows it as well. Take every opportunity to hold your knowledge over your partner's head. The best way to elevate yourself is to make your partner a little lower, even if only in your own mind.
2. **NEVER BACK DOWN:** Your opinion is correct, or else it wouldn't be yours. A correct opinion is worth fighting for to the bitter end. When one argument tactic begins to fail, switch to another, thereby flanking your opponent. Eventually, they will give up.
3. **GLOAT:** Remember that discussion from 15 minutes ago where your opinion gained the victory? Of course you do. Well, in case your partner has already blocked it out of his/her mind, bring it back up when you see a place where it applies. Say things like, "See how much better it is since we did this my way?"
4. **WATCH SILENTLY:** Nothing is more boring than watching someone code, right? Well, why make it better by communicating? Instead, sit silently and watch the code go across the screen. After a few minutes, take our your phone and update your status on Twitter or check your email. You may miss opportunities to discuss coding decisions, but you'll likely be able to gloat about them later when refactoring becomes necessary (which is inevitable since you're not in the driver's seat).
5. **CODE SILENTLY:** Why should you have to explain what you're doing in your code. Your pairing partner can see it on the screen! Besides, they probably wouldn't understand anyhow. Better to just code the way you always have because that's when you're at your best. Don't talk through your code and certainly never explain any decisions you're making.

6. **BETRAY TRUST:** When you're shooting the breeze with your co-workers or chatting on IRC, be sure to bring up funny or embarrassing stories from your "Pair Programming" experience. Of course, avoid subjects that reveal your own deficiencies, but be sure to highlight your pairing partner's lack or prowess. Doing this behind his/her back is encouraged, since you will likely feel more free to embellish.
7. **DOMINATE THE KEYBOARD:** Even though you are bent on killing "Pair Programming" in your team, you're not a complete tool. As the better coder, you feel responsible for getting the project complete despite the handicap the "agile crazies" have placed on your team. To that end, and working within the confines or "Pair Programming", you should make every effort to keep your position in the driver's seat.
8. **CODE ALONE:** Even though your team has decided to adopt "Pair Programming", it doesn't mean you can't sneak in some private coding time (for old time's sake). Ideal times to code privately are during coffee, lunch or restroom breaks. Be sure to start one or two sections of code while your partner is away from your workstation. As a result, you'll get more time in the driver's seat because you're the only one who knows your intent and direction.
9. **WIN:** Defeat is not an option for you. Stalemate is also not an option. It is imperative that you argue your point until your pairing partner sees your way of thinking or has laid down his guns and surrenders. One way to know you've won is when your partner ceases to express opinions and blindly accepts anything you say. One other benefit to this technique is that you will likely get to drive for the remainder of the project. Victory is sweet.
10. **COMPLAIN:** Chances are, your organization or team has one or two "agile crazies" that sold the higher-ups on the concept of "Pair Programming." That means the higher-ups are probably eager to hear about how it's going. If your other attempts at sabotage are effective, then you should have quite a bit to complain about. Go around your team mates and fire off some emails to your manager or his/her superiors. Be sure to explain the obvious reasons why two programmers on one keyboard simply cannot work. Explain that moral across the entire team is low and that productivity would be back to "normal" if they would just put you in a position of influence. Be sure to taint the reputation of the "agile crazies", maybe even calling them names like "agile crazies".

Above all, remember your greatness. You're better looking, smarter, more experienced, faster, and certainly more humble than anyone else on your team. Let those truths guide you, and, even though you'll be viewed by everyone around you as a jerk, you'll have single-handedly defeated the "Pair Programming" dragon and that should be enough to keep you warm at night… alone… with your cats.

---

From: http://www.awkwardcoder.com/index.php/2010/08/27/10-ways-to-kill-pair-programming/