

PAIR PROGRAMMING

For your assignments in this class, you are allowed to adopt a new strategy for software development: **pair programming**. This approach is very much at the forefront of software engineering practices today, and in many situations it has given impressive results. We feel it is especially effective when learning a new programming language and framework.

Pair programming: what it is

As the name suggests, if you adopt this methodology, you will work with a partner. There is, however, much more to it than that:

- All of the work on the assignment will be done by both team members working together at a single PC at the same time. No work is done individually.
- At any given moment, one person is at the keyboard and is known as the *driver*. The other person is observing and is known as the *navigator*.
- The driver and navigator collaborate on all aspects of the software development: design, coding, debugging, etc.
- The driver does the actual typing of the code.
- The navigator is thinking about broader issues: How does the current code fit into the larger picture? How will we test the code to see if it works? What should we do next? Are we making any discoveries that require changes to code already written? The navigator also observes the code being written and checks it for defects, but please understand that the primary role of the navigator is not to say "Oh, I think you are missing a semicolon there." This role is called the navigator because the job is to steer the software development in the right direction.
- The driver and navigator are in constant communication: asking and answering questions of each other and debating the best way to solve the problem.
- Periodically, the driver and navigator change roles. This is a crucial part of the methodology, and if you choose to do pair programming, you are required to spend roughly equal amounts of time driving and navigating.

Pair programming: why it works

It is easy to see the benefits of the paired approach when you are learning a new technology. As you have no doubt experienced, just "bouncing your idea off someone" is often very enlightening and very helpful. The act of explaining your solution helps crystallize it in your mind and helps you see any pitfalls. Add to that the fact that your partner will chime in with ideas of his or her own, and you can see that the power of two is much greater than the power of one.

Another benefit surfaces when you get stuck; that is, you arrive at some point where you don't know how to proceed or where your code is not working for reasons that are unclear. Sticking points like these can consume a few hours or even a few days.

Now imagine that you always have a partner to help you. In many cases you will find that where you are stuck, your partner will know the answer, and vice versa. Almost everyone who tries pair programming reports that there is far less "down time" due to programming difficulties.

Pair programming will also help you learn more. When your partner explains something to you, you learn something new. When you explain something to your partner, you learn it better. When you figure something out together, you both learn it, in less time than if you were working on your own. Mutual tutoring and problem solving is truly a win/win situation.

Pair programming: what it's not

Be sure you understand that pair programming does not mean dividing the assignment in half and each doing one part. That is one kind of software development team, but it is not what you will do in this class and it does not provide the benefits outlined above.

To reiterate: in pair programming, all the work is done with both team members at the same machine at the same time.

Pair programming: how to succeed

There are just a few of requirements for a successful pair programming project. In the first place, the partners must be compatible. Working with a friend is a really good idea; working with someone you don't like or normally disagree with is a bad one.

Another key ingredient is compatible schedules. Since you have to work at the same time, you have to be available at the same time. When forming a team, be sure to discuss your schedules and be certain that you have adequate opportunities for working together.

A final consideration has to do with ability levels. In any class, there is bound to be a range of programming abilities. Pair programming will work for you no matter where you are in that range, but it will work best if both partners are at about the same ability level.

This last point deserves some thought. You might believe that the best thing to do is find a partner who is a superstar, and that if you do, then all your worries are over. Wrong! Two things tend to happen in those cases: one person does most of the work, and the other person fails the exams. You have to exercise some self discipline and realize that to learn what you are supposed to learn from the assignment, you need to put yourself in a situation where the work will be shared more or less equally.

How we will evaluate your work

The basic idea is simple: you submit one assignment for the team, and both partners receive the same grade. We may ask you to submit a brief "peer evaluation" form, on which you rate the performance of your team, your partner, and yourself. This gives us confidence that the work was shared equally,

and if it appears that it was not, we may ask you to come in and discuss the way your team worked. In rare cases, we might end up giving different grades to the two team members.

Pair programming is not required

Pair programming is optional for this course. You may certainly work on your own if you wish, and all assignments are designed so that one person can complete them in the allotted time.

References

If you want to learn more about pair programming, a comprehensive discussion can be found in:

Pair Programming Illuminated, by Laurie Williams and Robert Kessler, Addison Wesley, 2003.

The notion of pair programming was popularized as part of the Extreme Programming (XP) methodology. A good reference is:

Extreme Programming Explained: Embrace Change, by Kent Beck, Addison Wesley, 2000.