Pair Programming Tutorial

This material is excerpted almost entirely from the webpage http://ecee.colorado.edu/~ecen2120/Manual/pair.html.

Background

Pair programming is a structured technique, with clear guidelines and methods, for improving the process of programming by using collaboration. Pair Programming is most effective when programmers have some basic understandings:

- Programmers should think of their work as a shared, collective project. Each programmer should be able to see his or her contribution in the final product, and each programmer should take equal ownership for the programming process that created that product: Each programmer should “own” both the process and the product of their shared work.
- Programmers should avoid the temptation to break projects into smaller parts to be completed independently and the integrated together. In pair programming, programmers work together, side by side, for most of the time. Pair programming aims to be truly collaborative.
- Programmers agree to perform specific activities while programming with their partners. Sitting side by side at one computer, programmers assume either the role of the Driver or the Navigator. Both of these roles are “programming”. The pair programming method depends on these set roles being carried out fully. Partners can switch roles when they wish and should work to feel comfortable with both sets of practices.
- Pair programming requires that partners communicate with each other openly and honestly. Yet, the Driver and the Navigator must work to balance their abilities to voice their own opinions while also being open to their partners’ input. This means staying away from a “my way or the highway” attitude. It also means having the ability to speak up and voice their own concerns. Pair programmers need to balance too much ego with too little.

Pair programming may seem to be more effort, or to be less efficient because it requires two people to work together, when most other programming practices are done with just one person. The efficiency of pair programming is shown in the outcome: generally reduced development time, generally better programs, and improved awareness of the logic and practices of programming.

Role Guidelines and Procedures

Effective pair programming requires some planning, and an understanding of the Driver and Navigator roles.

Set Up

Physical Space. The work space needs to be set up so that both programmers are able to sit next to each other, with both orienting to the computer. Programmers should have any materials on hand that they may need.

Planning Period. Before starting on their destination, programmers need to develop a mental map of where they are going. This includes discussing what they intend to do, their thoughts on the problem or design, and any questions or suggestions they may have at the start. At the end of this period, programmers have a shared sense of where they are going.

Role Assignment. Programmers must have an initial discussion to address who will navigate and who will drive.

The Driver

The Driver does the following:

1. Controls whatever is being used to record the program as it is developed, such as a pencil, mouse, or keyboard.
2. Has responsibility for the details of developing or writing the code. (We will refer to this as *entering code*.)
3. Talks out loud about what they are thinking about or why they are entering any particular element of the code. The intent here is to make reasoning or thought processes explicit so they can be discussed, rather than leaving them implicit and perhaps difficult to understand. This is one key to pair programming.
4. Responds constructively to feedback given by the Navigator. Recognizes that comments from the Navigator are part of the process, and doesn't become annoyed, defensive or dismissive. The Driver must be open to working out/talking through problems.

The Navigator

The Navigator does the following:

1. Does not enter the code. This is important. Partners should make every attempt to talk through the ideas rather than simply hand off the keyboard or pencil to each other. Hand-offs prevent truly understanding the program as it is being developed.
2. Keeps an objective view and thinks strategically about the direction in which the program is going. Navigators keep a big picture view.
3. Watches and alerts the Driver if there are tactical or strategic problems with the work.
4. Continues to think of alternatives and looks up resources that may help the partners pursue other ways of going about solving a problem. These are things the Driver shouldn't do while focusing on developing code, yet could be critical to developing a good program.
5. Looks for the strategic implications of the developing code. Asks questions at the appropriate times, such as: where are we going, and what does it mean to be doing it this certain way over others?
6. Asks questions of the Driver, especially if the Driver is doing something that is unclear, cloudy, or unknown to the analyzer. Asks these questions in constructive and supportive ways, rather than in ways that are directing or commanding. Listens to the Driver's answers and discusses if appropriate. These include: Try this ..., I notice that ..., etc.

Procedural Checklist

Before Entering Code:

- Physical space is correctly arranged and materials needed are present.
- Discussed what you are working on and who is doing what.

Driver:

- Should be the only one entering code.
- Talks through their actions.
- Accepts feedback and direction from the navigator.

Navigator:

- Focuses on the big picture.
- Thinks strategically–is watchful and alert to what the Driver is doing.
- Shares alternatives and consulting resources when needed.
- Asks questions of the Driver.

Programmers:

- Both are listening and responding to each other.
- Neither programmer is holding back their opinion(s) or question(s).
- Criticism/suggestions are made constructively.
- Both programmers own the project.