Pair Programming

CS 346 Application Development

https://student.cs.uwaterloo.ca/~cs346

Contents

Introduction

What is Pair Programming?

Pair programming is a software development practice where two people work together at a single computer, collabrating to solve a problem.

This was originally an Extreme Programming (XP) practice.

- The motivation was to reduce the number of defects, by having two people actively work on design/implementation together.

Pros/Cons

- “Two sets of eyes” on the problem actually does improve the quality of code.
- The downside is obviously that you’re doubling up the effort being applied to a problem.

While a pair of developers work on a task together, they do not only write code, but they also plan and discuss their work. They clarify ideas on the way, discuss approaches and come to better solutions.

What is Pair Programming?

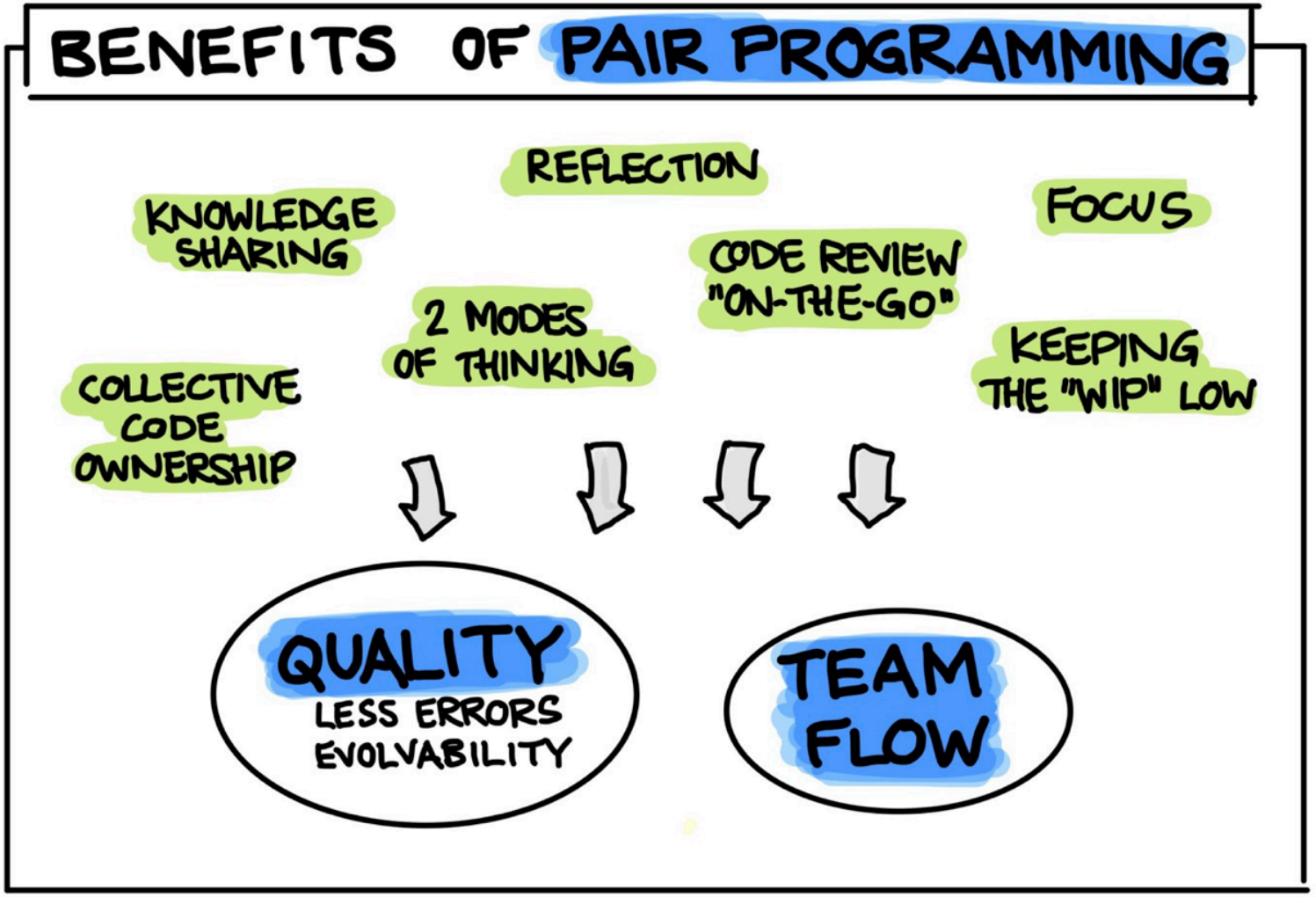


Figure 1: Pair programming has many benefits beyond improving code quality [1].

Guidelines

Physical Setup

Ideally, a pair works together in the same space.

- Make sure that both of you can sit facing the computer.
- Agree on the computer setup (key bindings, IDE etc).
- Check if your partner has any particular needs (e.g. larger font size, higher contrast, …)

Remote Setup

There are also development tools that are designed to specifically support remote sharing.

- JetBrains Code With Me
- Visual Studio Live
- CodeShare

Let’s discuss how this will actually work!

Pairing Styles

Driver and Navigator Pairing

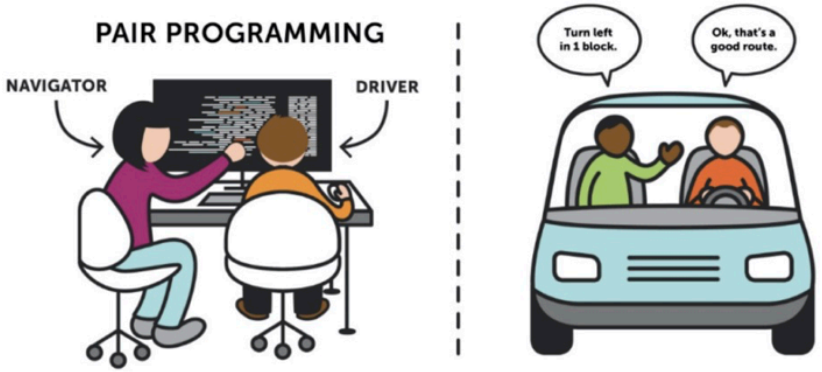
Classic form of pair programming.

The `Driver` is the person at the wheel, i.e. the keyboard.

- Focused on completing the goal at hand
- Should talk through what she is doing.

The `Navigator` is in the observer position.

- Reviews the code on-the-go, gives directions and shares thoughts.
- Watches for larger issues, bugs
- Makes notes of potential next steps or obstacles.
- The navigator should have a pad of paper (not a cell phone).



Driver and Navigator Pairing

A common flow goes like this:

- Start with a reasonably well-defined task. Pick a requirement or user story.
- Write down your goals and keep track of them (paper, whiteboard).
- Agree on one small goal at a time. This can be defined by a unit test, or by a commit message, or some goal that you've written down.
- Discuss the overall design as a team. The driver then implements it while the navigator watches.

As the navigator, you oversee medium-term thinking and planning.

- Provide feedback on the immediate goal. Anything else should be written down for afterwards.

As the driver, focus on the immediate goal.

- You are coding.
- Explain what you are doing to the navigator. Ask for feedback if needed.

Ping-Pong Pairing

This technique is ideal when you have a clearly defined task that can be implemented in a test-driven way.

- “Ping”: Developer A writes a failing test
- “Pong”: Developer B writes the implementation to make it pass.
- Developer B then starts the next “Ping”, i.e. the next failing test.
- Alternate over and over.

Each “Pong” can also be followed by refactoring the code together, before you move on to the next failing test.

> **Tip**
>
> This is a great style if you're using TDD. It's a suitable alternative to classic pairing in this course if you wish.

Strong-Style Pairing

In this style, the navigator is much more experienced (with the language, the tool, the codebase).

- Our goal is knowledge transfer!

Navigator

- The more experienced person.
- Guides the driver, and explicitly tells them what to do.

Driver

- The less experienced person
- They take explicit direction from the navigator.

The driver should trust the navigator and be “comfortable with incomplete understanding”. Changes are discussed afterwards.

> ⚠ **Warning**
>
> This style is not suitable for a project course like this. Use another one.

Bibliography

[1] B. Böckeler and N. Siessegger, “On Pair Programming.” [Online]. Available: https://martinfowler.com/articles/on-pair-programming.html