
Pair Programming
Bob Fornal, Schamir Poliard, and Luis Carmona

About Me

- Husband and Father
- Senior Solutions Developer at Leading EDJE, Inc.

- 1978 - Started Writing Code
- 1990 - Completed BS CPS

About Schamir

- Technologist at Strategio
- Content Creator of ZONA
- Community Manager
- Interests in Developer Advocacy

About Luis

- Technologist at Strategio
- Built my first PC
- Started writing code
- Co-host of The Boba and Cherry Cafe twitter space.

Why Pair

- On teams, developers want to get more done.
- Less experienced developers need to learn from the more experienced developers.

"Pair programmers: Keep each other on task. Brainstorm refinements to the system. Clarify ideas. Take initiative when their partner is stuck, thus lowering frustration. Hold each other accountable to the team's practices. Pairing" - Kent Beck

Pairing allows developers to ...

- Produce better solutions.
- Share knowledge and context on the fly.
- Mutual learning and skill development.

Definitions

Pair Programming is the practice of pairing up to work on development tasks.

Most people think of two developers sitting at the same machine, sharing a keyboard. With the increasing popularity of remote work, it is now possible to pair program while thousands of miles apart.

Definitions

Driver

- Fingers on keys; typing code.

Navigator

- The developer that indicates what should be typed; that decides the direction to go.

Pairing

- Should become **Habitual**.

- Improves **long-term velocity** (for a short-term drop).
- Allows viewing the code in **Multiple Levels of Detail**.

Pairing Awareness

- Partner’s voice / body language (how are you/they feeling).
- Parter’s emotional state (excited or defensive).

Pattern: Strong Technique

The Strong Technique is an approach where the driver does nothing that the navigator did not direct.

For the driver to provide an idea, they must hand over the system to the partner and then carry out the control from the navigator position.

Pattern: Traditional Technique

Traditionally, pair programming has been popularized through the driver-navigator technique. Ideally both the team members should switch roles at times for better results.

With this style of pair programming, the navigator often loses interest.

Pattern: Ping Pong Technique

(TDD Swapping)

Using this pattern, one of the developers writes the test that fails and the other developer writes the code to get the test to pass. Then, the pattern is repeated.

The biggest challenge with this pattern is finding time refactor the code and do testing.

Pattern: Leap-Frog Technique

This pattern can be best described by looking at two developers in different time-zones. One is working on front-end code, the other on back-end code. They work toward a common feature, taking the development lead ... moving the code forward.

Pattern: Rubber Duck Technique

(Grab a Buddy)

Here, the pairing is often two developers at similar levels.

One, working on some code, is stuck and in need of an "extra pair of eyes." Often, someone not as "deep in the weeds" will be able to see what the original developer missed.

Pattern: Mob Technique

The Mob Technique can be used well with code abstractions to develop a conceptual implementation.

This technique allows several developers to contribute at one time. Many are referencing code and finding implementation patterns for use or validation, while one or two individuals are writing code.

This technique could also be used when in a teaching (or mentoring) role to give the group room to explore and examine code patterns.

Tools

The big problem is not seeing the code, but interacting in a meaningful way. Being able to select code or highlight a section for clarity is important.

- VS Code: Live Share
- Zoom: Make Co-Host and use Annotate Tools.
- Slack Huddle: Sharing and Drawing.

Have line numbering turned on so that the conversation is, “Let’s take a look at line 93.”

DEMO

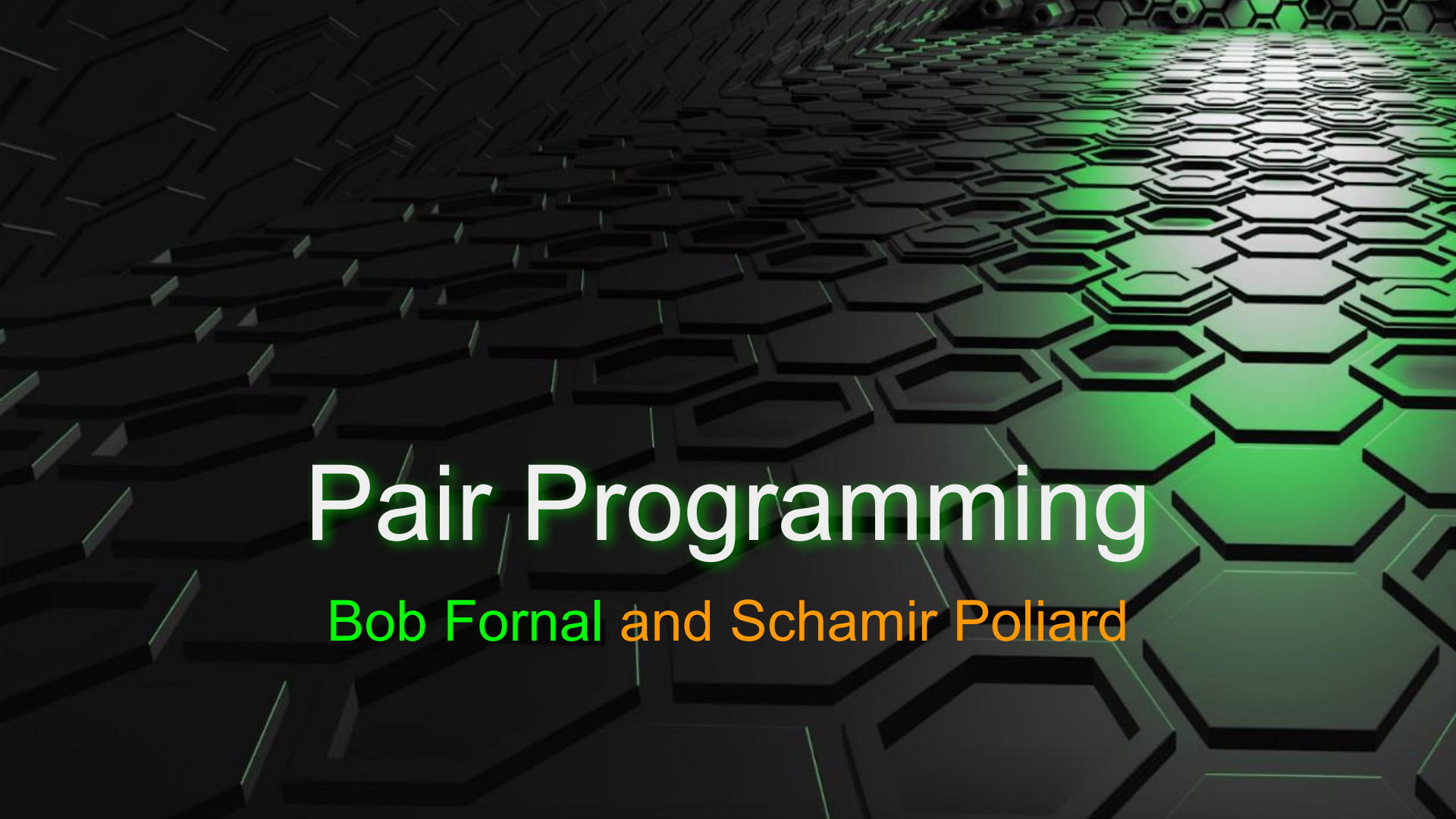
Summary

- Strong Technique
- Traditional Technique
- Ping Pong Technique
- Leap-Frog Technique
- Rubber Duck Technique
- Mob Technique

Conclusions

Pair programming, when done correctly is a powerful tool, but it can easily go wrong and ...

- Should never be forced.
- Should not expect different results or speed than traditional development.
- Is not about "showing off."


Pair Programming
Bob Fornal and Schamir Poliard