

# Pair Programming

Fundamentals and aspects to have in mind



# Fran Ávila

Software Craftsperson

[fran.avila@codurance.com](mailto:fran.avila@codurance.com)

@Ripham\_78



## Classic Pair Programming

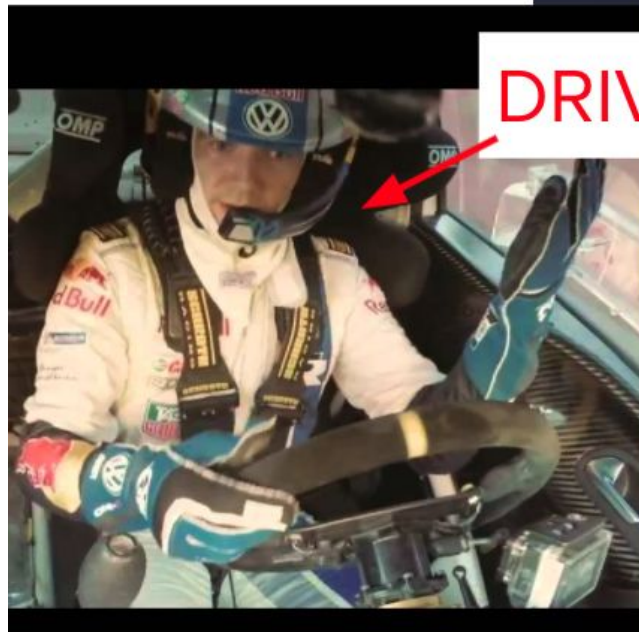
NAVIGATOR

DRIVER



## Driver

- Operating the keyboard
- Tactical: focuses on the code at hand (the syntax, semantics, and algorithm...)
- should be talking constantly
- Talks out loud about what they are thinking about or why they are entering any particular element of the code
- Responds constructively to feedback given by the Navigator





## Navigator

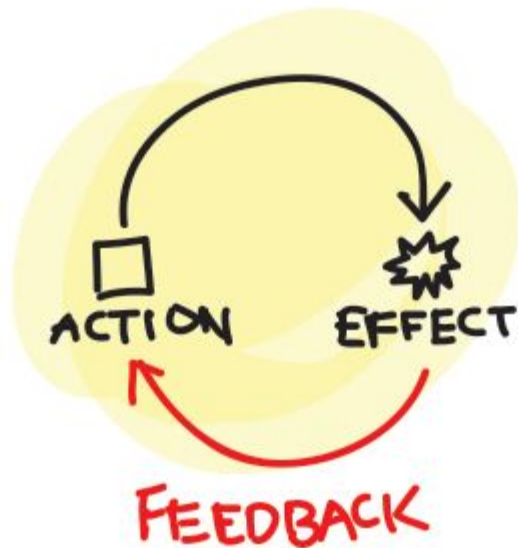
- Big Picture
- Strategic: focuses on a level of abstraction higher (test they are trying to get to pass, the technical task to be delivered next, the time elapsed for test, for commit, and the quality of the overall design...)
- Continues to think of alternatives and looks up resources
- watching, alert, learning, asking, talking, and making suggestions
- Asks questions of the Driver

NAVIGATOR



## Benefits

- Increase quality
  - Homogeneous code
  - Knowledge transfer
  - Focus
  - Risk reduction
- 
- Reduce feedback cycle time



## Pair Programming Scenarios

TYPE	DRIVER'S KNOWLEDGE	NAVIGATOR'S KNOWLEDGE	SCENARIO
<ul style="list-style-type: none"><li>• Non structured</li><li>• Classic</li><li>• Ping-Pong</li></ul>	High	High	Best
Guided tour	High	Low	Bad
Driving Instructor	Low	High	Good
WTF are we doing????	Low	Low	Low productivity, High learning

# Some thoughts

- Is a **social skill** and it takes time to learn it
- Best are the one who said “let’s try your idea first”
- Is not **mentoring** (even though can be used for it)
- Takes time. At first, it’s **awkward**; this is normal
- Individual 30 - Pair 50 - More 20 (Peopleware, Tom DeMarco 1987)
- Require **discipline**
- The **constant feedback** is a must

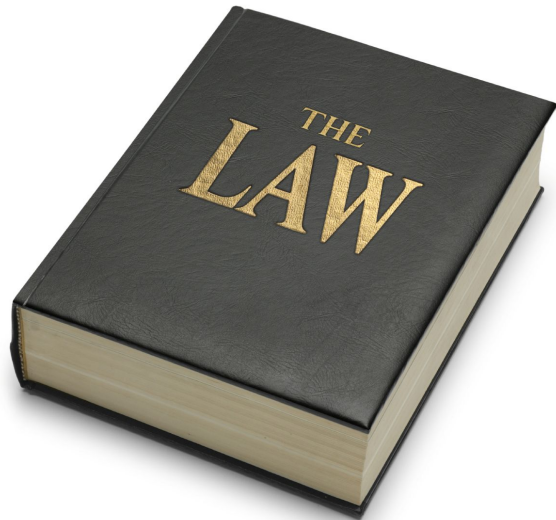
“**Don’t worry about the** Pair Programming: You are not as good as you think, and you are not as bad as you fear.





## The LAW

“For an idea to go from your head into the computer it **MUST** go through **someone else's** hands”



DOs	DON'Ts
Agree the environment (physical) before to start	Express superiority
Have common agreements, principles, etc. before to start	Never retract
Talking about code, be explicit	Stay in silence
Determine the starting point, the goal, and what means “better”	Betray trust
We are humans with feelings. Be assertive, respectful and clear.	Over control the keyboard
Give as many compliments as you can	Code alone
Be ready to “lose”	“Win” no matter what
Over-communicate	Whining
Wait and accept corrections	All or nothing situations
Give space	
Allow different speeds	
Change role frequently	
Build a relationship	
Get frequent breaks	



## Mob Programming

- **Whole** team aligned
- Homogenic code. Common rules.
- Learning and sharing knowledge
- Less meetings
- Less communication problems
- Faster decision making

Driver/Navigator

Rotate  
Every 15 minutes

Driver



<https://www.youtube.com/watch?v=Ev7uus12HRY>

<https://www.agilealliance.org/resources/sessions/mob-programming-aatc2017/>

## Pair Programming Remoto

- Use tools like Code Together, LiveShare, Wemux, etc
- Have a video call with chat
- Use shared documents
- Over-communicate even more
- Stop more frequently
- Check your partner status
- Obicuous language
- Book time to be alone









**T.HANKS**