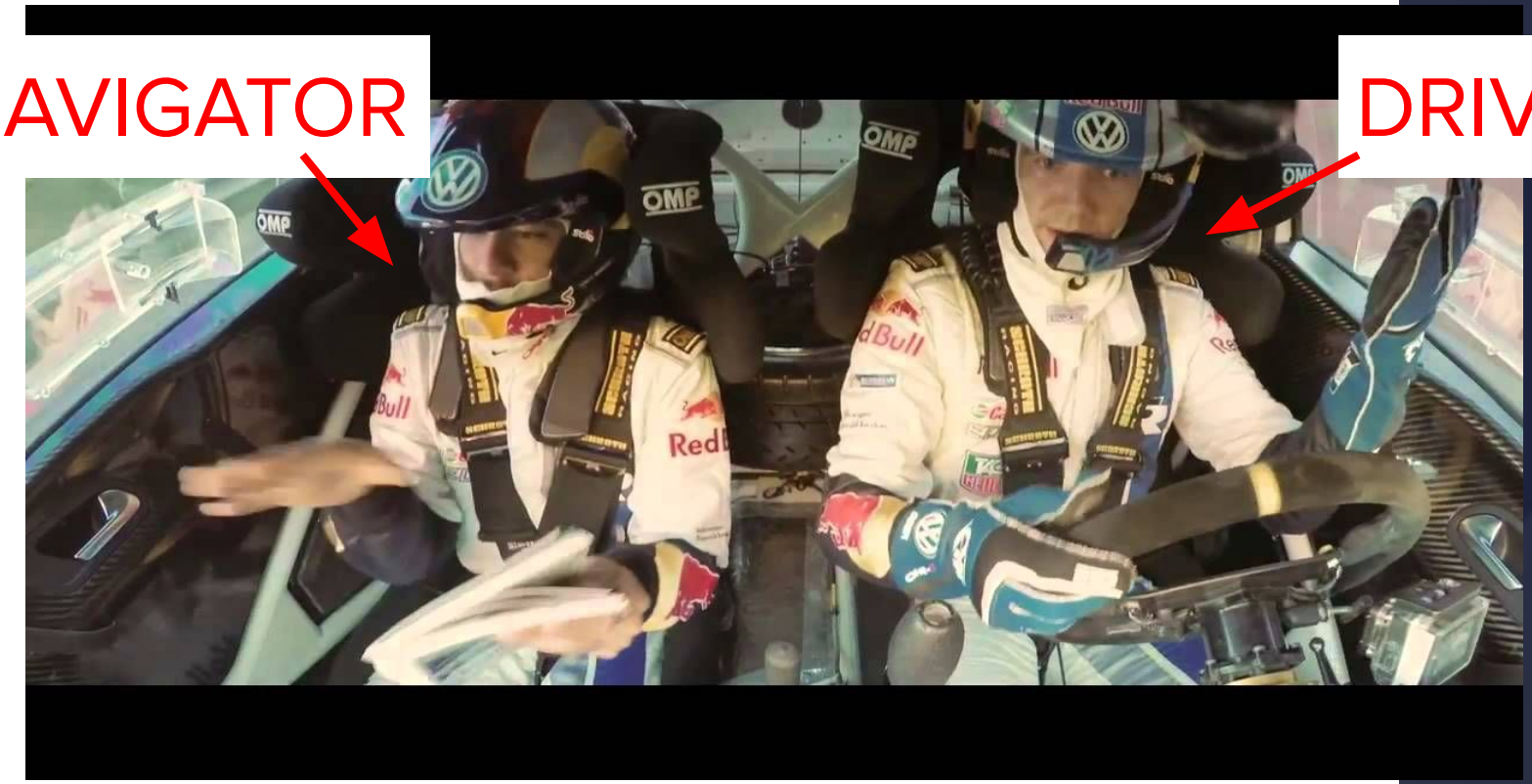
codurance.com

Pair Programming

Fundamentals and aspects to have in mind

codurance

Classic Pair Programming



Driver

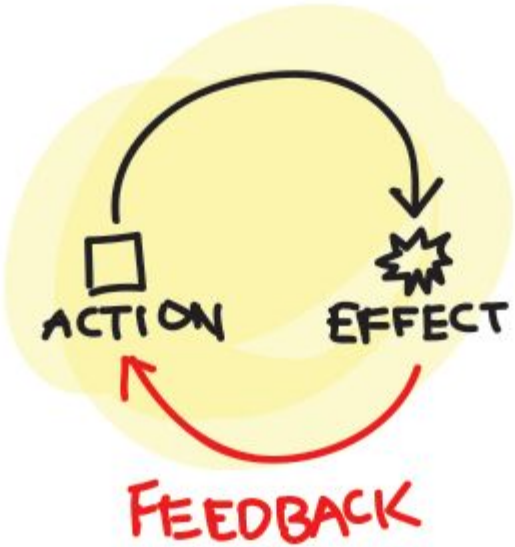
- Operating the keyboard
- Tactical: focuses on the code at hand (the syntax, semantics, and algorithm...)
- should be talking constantly
- Talks out loud about what they are thinking about or why they are entering any particular element of the code
- Responds constructively to feedback given by the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

Navigator

- Big Picture
- Strategical: 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

Benefits

- Increase quality
- Homogeneous code
- Knowledge transfer
- Focus
- Risk reduction

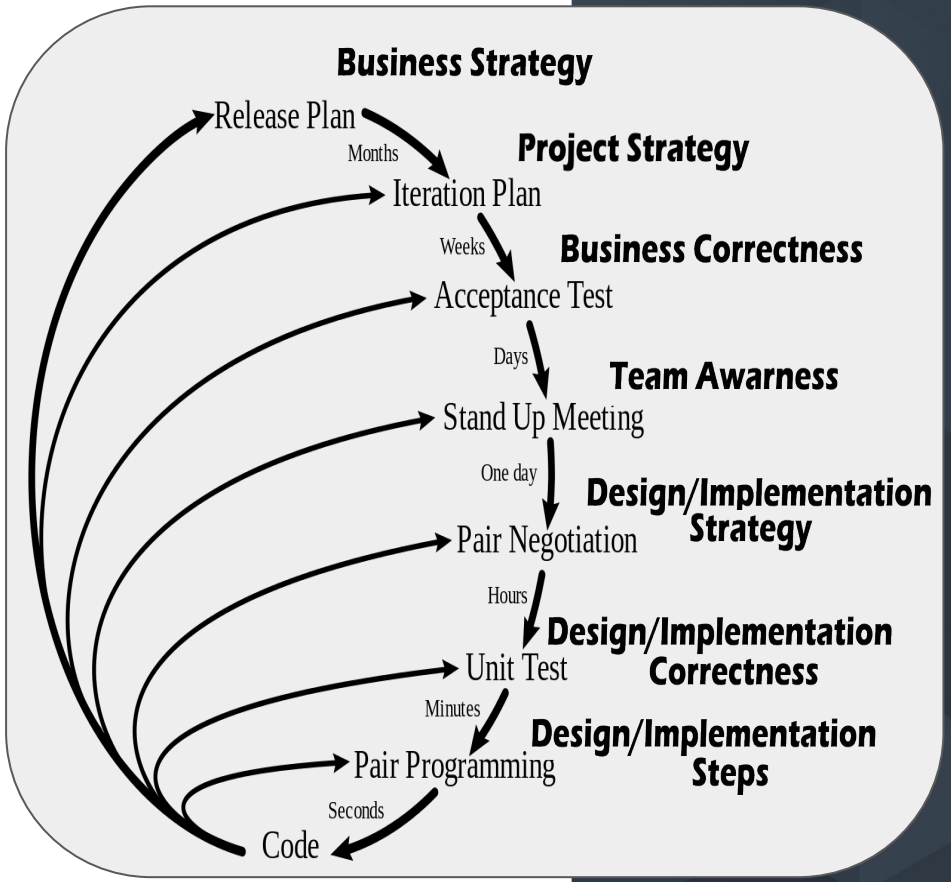


- Reduce feedback cycle time

"Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, courage, and respect. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation"

"Optimism is an occupational hazard of programming: feedback is the treatment"

Kent Beck



“The adjustment period from solo programming to collaborative programming was like eating a hot pepper. The first time you try it, you may not like it because you are not used to it. However the more you eat it, the more you like it.”

Anonymous XP Practitioner

Pair programmers:

- Keep each other on task.
- Brainstorm refinements to the system.
- Clarify ideas.
- Take initiative when the partner is stuck, thus lowering frustration.
- Hold each other accountable to the team's practices.

Driver/navigator switch techniques

Both partners should spend an equal amount of time in each role. How?

- **Chess clock**
  - Roles switch after a predefined time interval
- **Ping pong**
  - Dev A writes a new RED test. Dev B writes code to pass the test. Now Dev B writes a new RED test. Dev A writes code to pass it. And so on.
- **Pomodoro**
  - Set a pomodoro timer at x minutes and when it rings take a short break and switch. After 4 pomodoros take a longer break.

Driver/navigator switch techniques

Asymmetric pairing

This practices can be useful in some contexts of different skill levels (senior/junior).

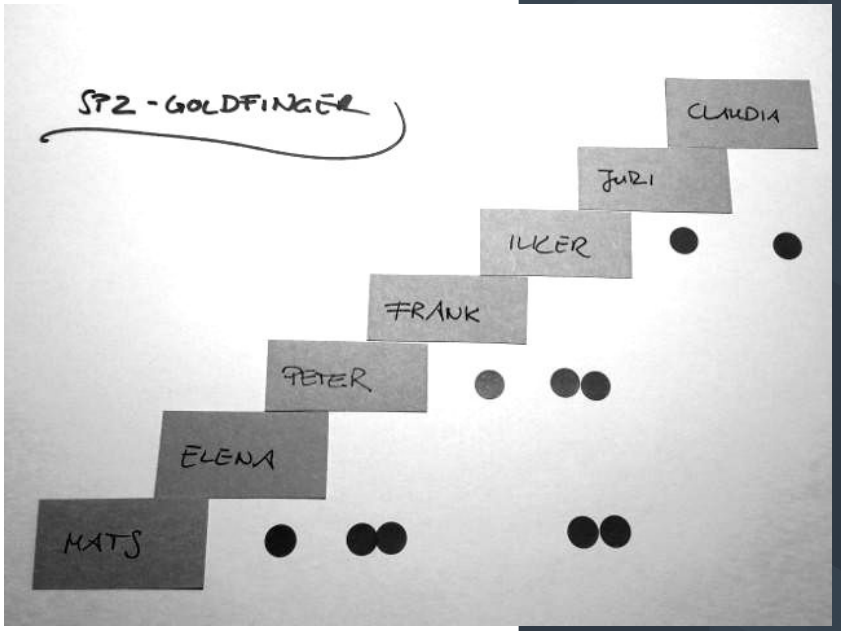
- **Guided Tour**
  - Senior drives and navigate.
  - Junior asks questions and learn from what's happening.

- **Driving Instructor / Backseat navigator**
  - Junior drives.
  - Senior navigates and tells the drives all the details.

Pair rotation

It is important in a team that all pairing combinations work together for some time without too much imbalance.
To keep track of this the “pair stair” board can be used.

- Whenever two team members pair up, they mark their “pairing” with a dot.
- The goal is to fill the voids, which are the pairs who have not yet worked together.



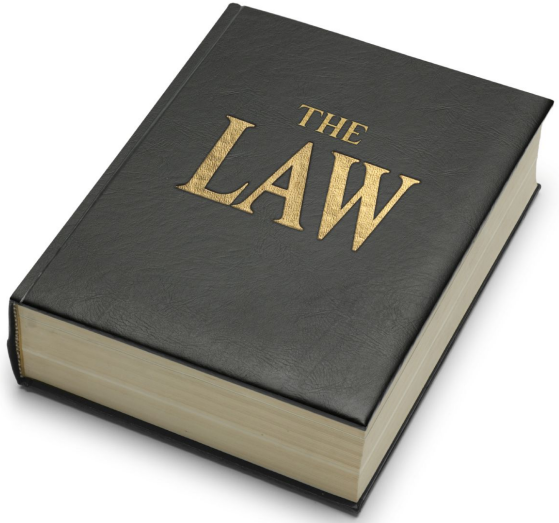
TYPE	DRIVER'S KNOWLEDGE	NAVIGATOR'S KNOWLEDGE	SCENARIO
• Non structured • Classic • Ping-Pong	High	High	Best
Guided tour	High	Low	Bad
Driving Instructor	Low	High	Good
WTF are we doing????	Low	Low	Low productivity, High learning

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

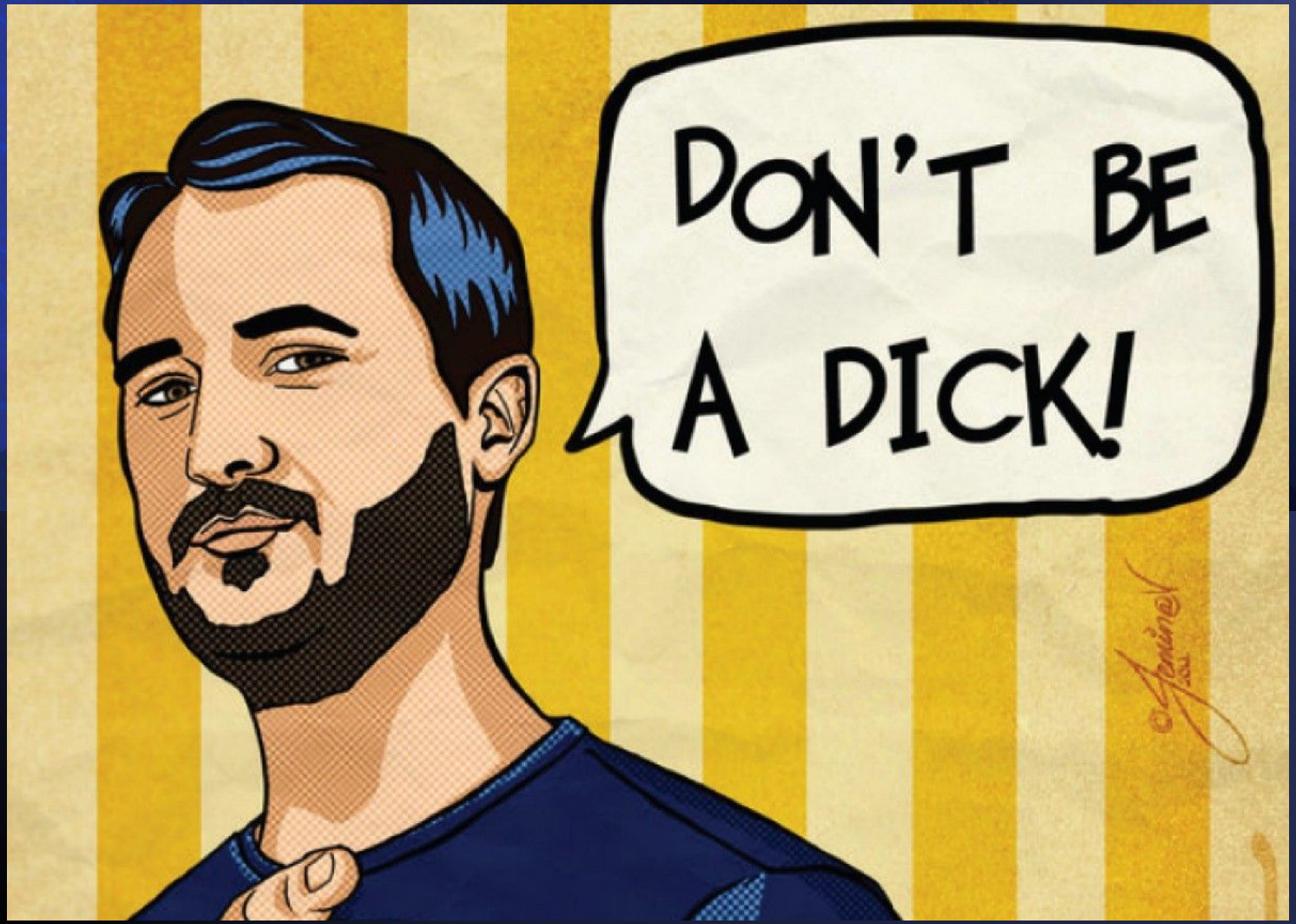
“**Don’t worry about the** Pair Programming: You are not as good as you think, and you are not as bad as you fear.

The LAW



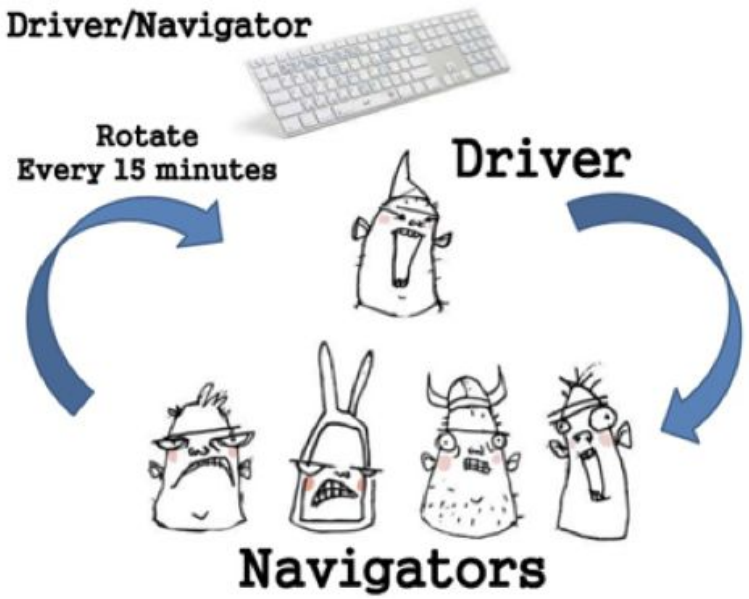
“For an idea to go from your head into the computer it **MUST** go through **someone else's** hands”

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	


DON'T BE
A DICK!

Mob Programming

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



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

Pair Programming
Q & A


T.HANKS