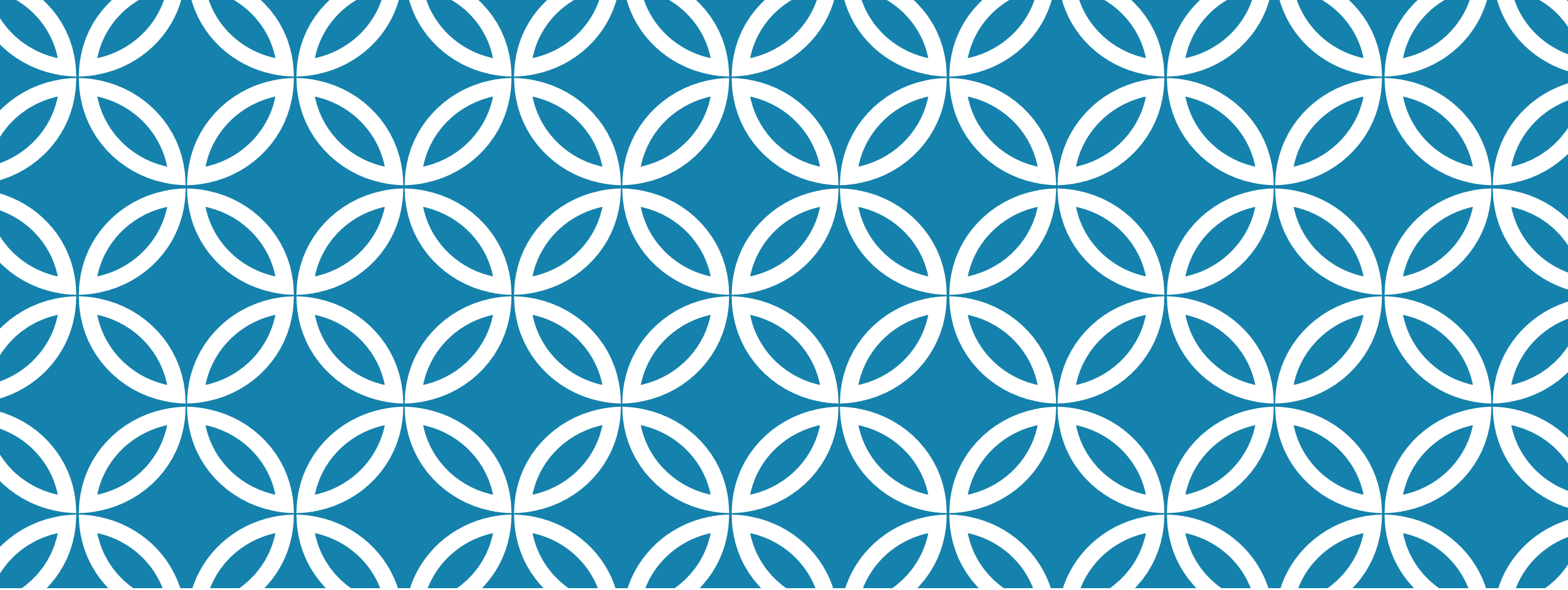


PAIR PROGRAMMING

Nov. 27

ANSWER THESE QUESTIONS:

- What is pair programming?
- One pro
- One Con
- Overall opinion

- your interest from 1-10?

WHAT IS PAIR PROGRAMMING PATTERN?

- Two developers
- Share the same workstation
- One is the Driver
- The other is the Navigator, that helps the Driver with directions
- Switch the roles
- On one computer only
- Two sets of keyboards, two mice
- NOTE: it is not pair-coding!

EFFECTIVE FACTORS

- When to pair and when not to:
  - Not for every tasks; e.g. simple tasks
  - For a difficult task
  - For touching a core feature:
    - having a senior – junior combination, opportunity to strengthen the junior skills and knowledge
  - For debugging – saves a lot of time
- Type of person matters!
- Should try it out to find out!
- Don't overdo it! – Pivotal labs

PAIR COMBINATION

- Skill level of the developers:

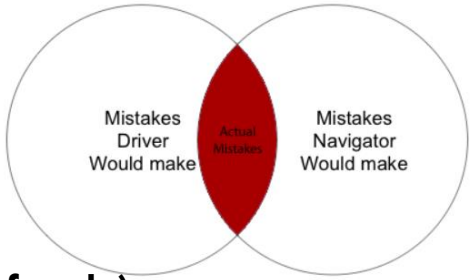
	Execution Work	R&D Work
Senior + Junior	GOOD	OK
Junior + Junior	GOOD	NOT GREAT
Senior + Senior	OK	NOT GREAT

- The personality of the developers

	Execution Work	R&D Work
Passive + Dominant	GOOD	OK
Passive + Passive	OK	NOT GREAT
Dominant + Dominant	NOT GREAT	NOT GREAT

THE WHY

- **Fewer mistakes:**



- **Code quality (even fewer lines of code)**

- **Easier to Keep Going**
  - more positive about trying again and again
- **Harder to Procrastinate**
- **Shared Best Practices and tips**
- **Faster On-Boarding**
- **Identify and Reduce Bad Hires**
- **Increase Employee Satisfaction – not necessarily!**
  - **Enjoy (soothing to be sure that that no major mistakes)**
  - **Confidence**
- **makes people replaceable**
- **Less context switch cost**

THE WHY NOT

- **Tired Coders**
  - For some people
  - No downtime
- **Tooling alters productivity**
  - color selection
  - keyboard positioning
  - screen height
  - font selection
- **Social overhead**
  - Not feeling confident
  - Can be distracting
- **highly social and interactive**
  - Not working on every character
- **On short term execution, sounds to be twice as expensive?**
- **Or Half the throughput?**

SOME PEOPLE IDEAS

- Someone's:
  - For debugging – so efficient
  - Remote pairing – effective
  - Enforced pairing – with good and bad points
    - Nothing to hide
    - No privacy
    - Too quick
  - Mob programming

- interrupt my stream:
  this is because it is got as pair-coding, not pair-programming!

- Remote is also a problem
  I personally did it

- Some people like programming, because they want to work alone!

PRACTICES

- For remote workers:
  - Online screen sharing
- The programmers should be vocal and think loud
  - Specially the driver
  - Cheer up
  - Also, be a good listener
- When to switch:
  - *Pomodoro – using a timer*
  - *Ping pong – one test, other resolve and reverse*
  - *Switch on google*
  - *Task list*
- *10-seconds rule*

CHALLENGES

- Continuous social interaction
- How to manage when there is gap
  - Mentoring?
- code ownership
- Task ownership
- Reward Shared Contribution
- Convincing managers! :D
  - Development cycle
- When driver does not think loud
- When navigator just watch
- When driver does not listen to/trust the navigator
- When do you go to the bathroom?
  - Reduce intensity

SUMMARY

- social skill

- matter of finding the right combination of developers for the right task

- 15% slower, 15% fewer bugs

"Using interviews and controlled experiments, the authors investigated the costs and benefits of pair programming. They found that for a development-time cost of about 15%, pair programming improves design quality, reduces defects, reduces staffing risk, enhances technical skills, improves team communications and is considered more enjoyable at statistically significant levels."

+ work on social development

+ have to take shower more often!!

+ Bonus points: some non-work chat

- how do you think to keep a matrix of different pairs?

AGAIN

Overall opinion?

- From 1-10?