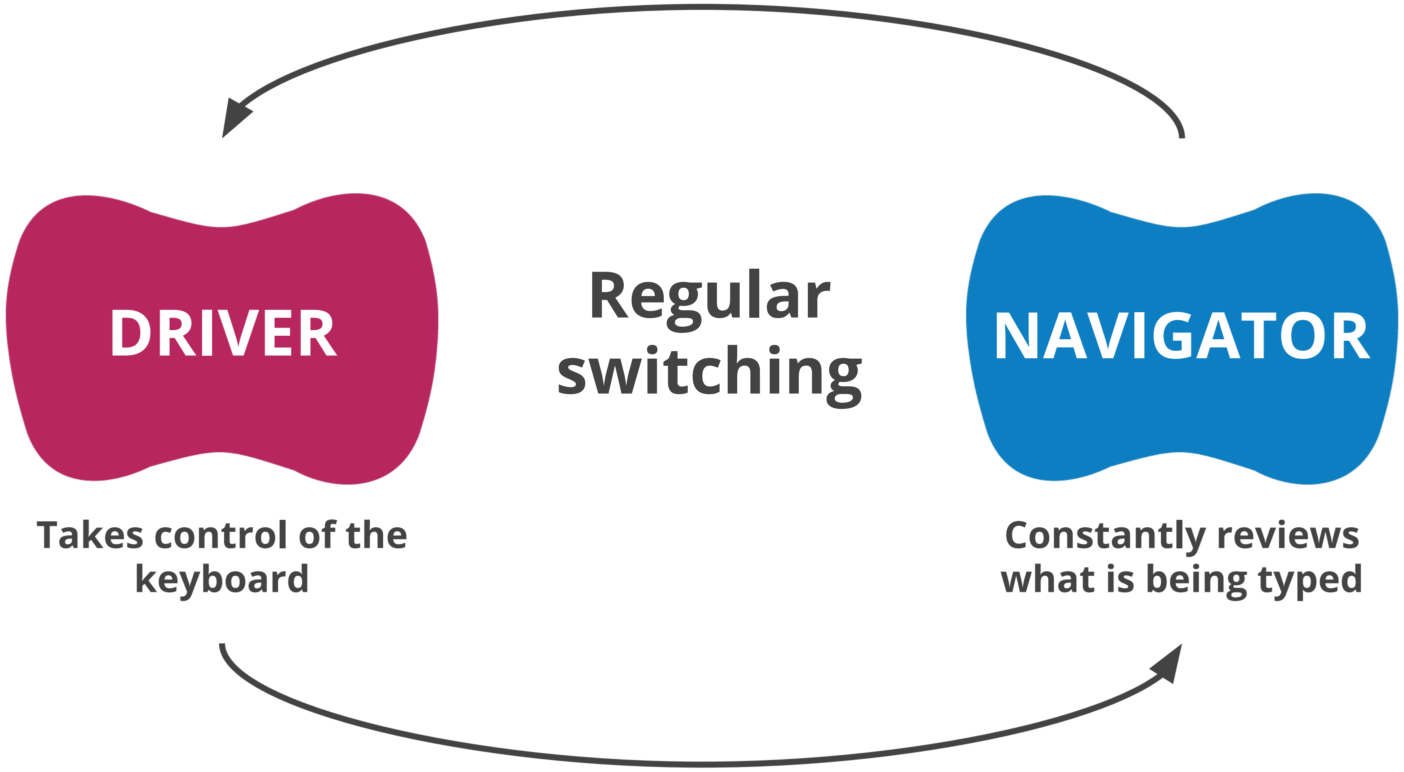
ThoughtWorks®

LevelUp Build

PAIR PROGRAMMING AND TEST-DRIVEN DEVELOPMENT

PAIR PROGRAMMING ROLES


DRIVER
Regular switching
NAVIGATOR
Takes control of the keyboard
Constantly reviews what is being typed

EXPECTED BENEFITS

- **Continuous Code Reviews.** Many mistakes get caught as they are being typed in rather than in QA test or in the field, leading to lower end defect count

- **Pair Relaying.** The designs are better and code length shorter

- **Line-of-sight learning.** The people learn significantly more about the system and about software development

EXPECTED BENEFITS

- **Shared Understanding.** The project ends up with multiple people understanding each piece of the system

- **Improved Collaboration.** The people learn to work together and talk more often together, giving better information flow and team dynamics

- **More enjoyment.**

CORE PAIRING GUIDELINES

1. **Collaborate; don't critique**
2. Actively contribute
3. Switch roles frequently
4. Find comfortable work stations
5. Rest if you must.

PAIRING SMELLS

1. **Unequal access**

   e.g. one person dominating the keyboard

2. **Unhealthy relationship**
3. **Worker-rester relationship**
4. **Endless debate**
5. **“Go make a cup of tea”** syndrome

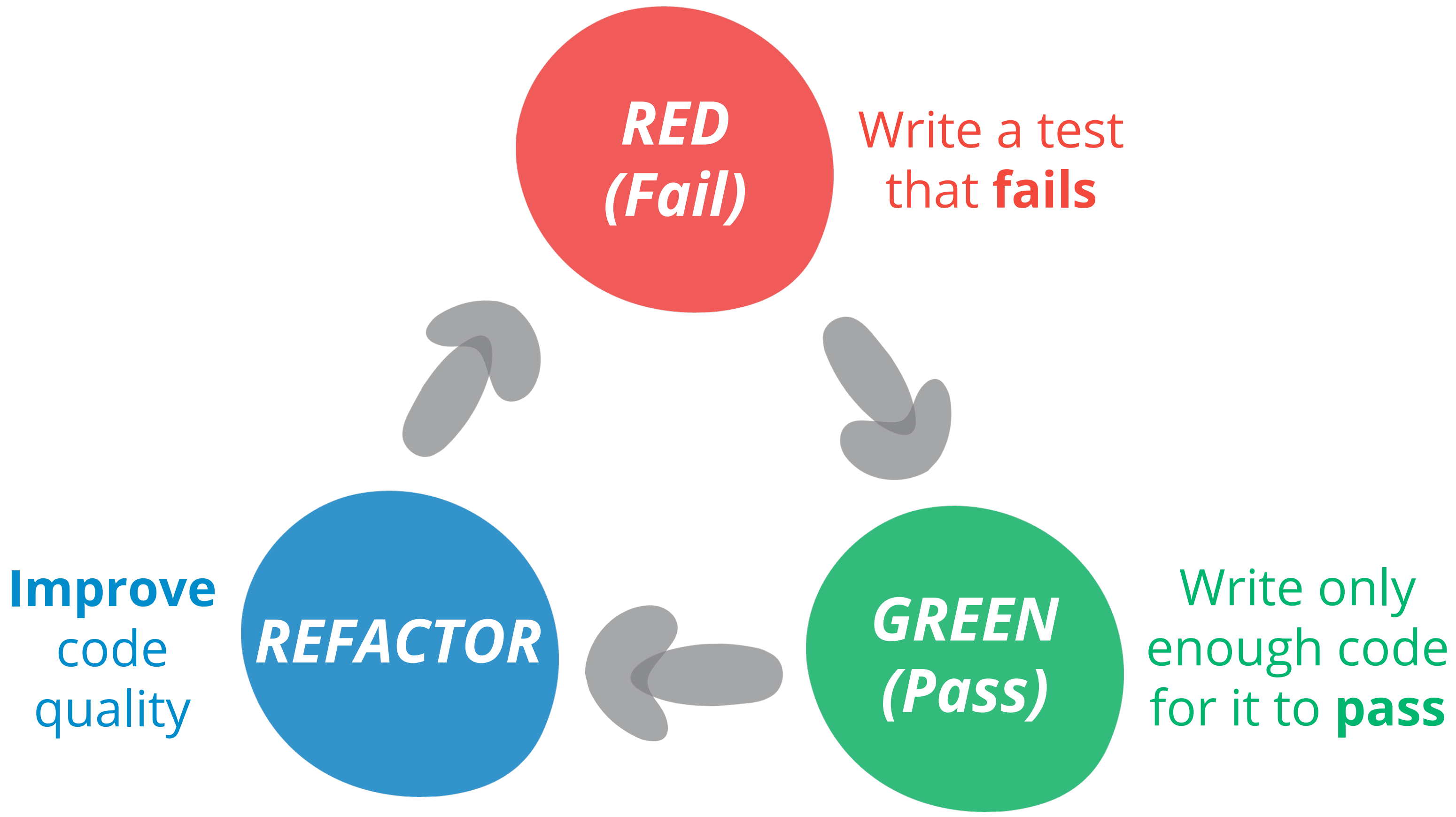
WHEN IS PAIR PROGRAMMING LEAST EFFECTIVE?

1. **When** working on non-complex mechanical-like tasks
2. **When** both parties do not have the same level of expertise - great for training but pairs are more engaged when they have the same level of expertise
3. **When** pairs don't rotate

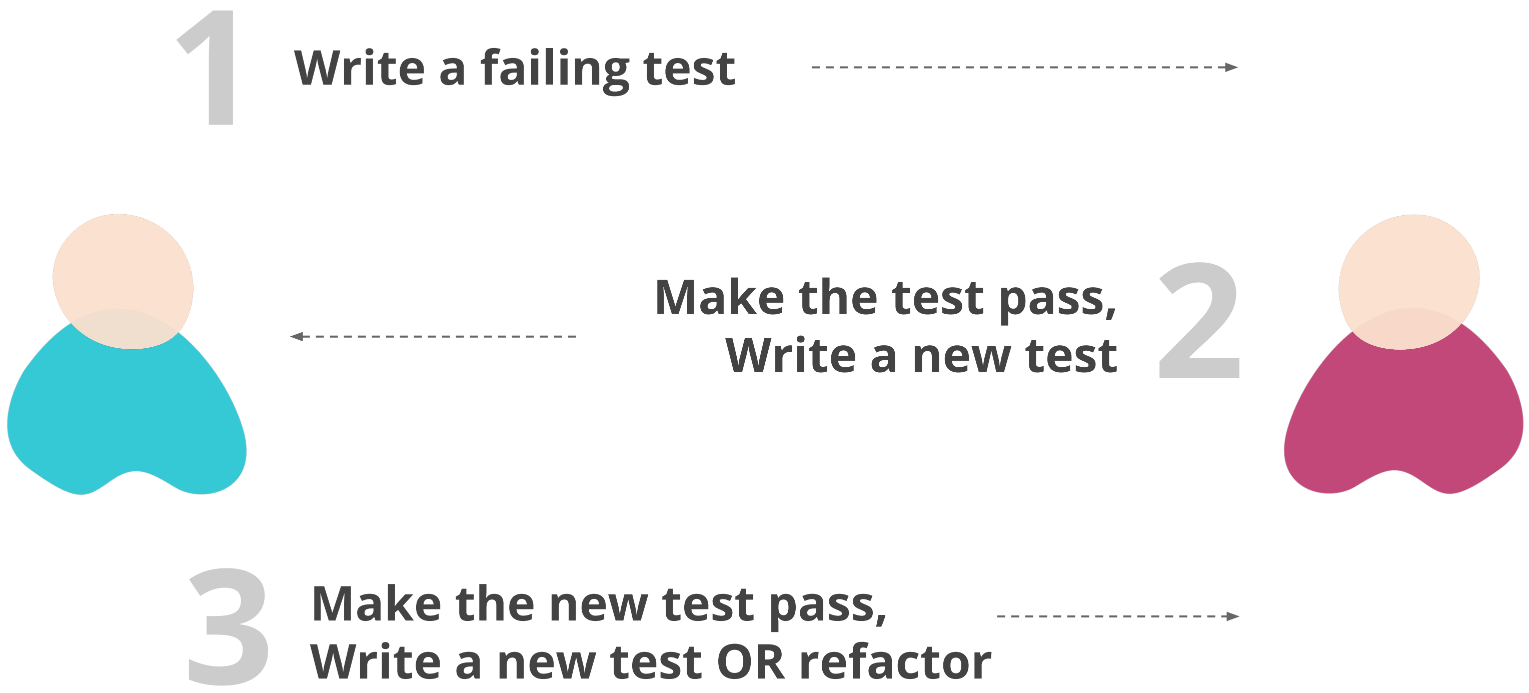
ThoughtWorks®

TEST-DRIVEN DEVELOPMENT (TDD)

RED-GREEN-REFACTOR: THE TDD MANTRA



TDD + PAIR PROGRAMMING (PING PONG METHOD)



1 Write a failing test

2 Make the test pass,
Write a new test

3 Make the new test pass,
Write a new test OR refactor

(continue back and forth with step 3)