Increasing Your Team’s Bus Count

Sustainable Software Development through Overlapping Pair Rotation

Authors
Todd Sedano
Paul Ralph
Cécile Péraire

(photos licensed or ©Todd Sedano)
Learning Outcomes

Participants will develop a proactive strategy for distributing knowledge throughout the team. Teams might pretend that disruptive events are not going to happen to them, but that’s not a reality in software development.
Why?
1. Team losing information

2. Learning curve for new developer
Knowledge
We want to spread knowledge throughout the team
Agenda

Motivation
Activity - Current state of team knowledge
Activity - Removing knowledge silos
Review Pivotal’s Process
Activity: who has what knowledge?

First, we want to see where the knowledge is on the team
Activity: who has what knowledge?

Individually, describe each team member’s knowledge
Debrief
In order to minimize impact of disruption, we minimize knowledge silos
Activity: removing knowledge silos

Individually, list strategies for moving knowledge around team
Activity: removing knowledge silos

Select two you think would help the most with moving knowledge
Activity: removing knowledge silos

Discuss your activity with your partner

What could go wrong?
How would you know if it was working?
How would you “sell” it to your team?
Debrief
Research and Project Quattuor
Research Context: Pivotal

**Mission:**

a) Deliver highly-crafted software  
b) Provide a transformative experience for client and combine client engineers with pivotal engineers

**Project team sizes:** 2 to 28 devs (6 is common)

**Balanced team:** interaction designer, product manager, devs

**Method:** Extreme Programming
Original Staffing Plan

Developers Allocated

Week # of the Project
Actual Staffing Plan
Theory of Sustainable Software Development
Mitigates negative disruption effects by
1. Spreading knowledge around the team
2. Writing “well-factored” code
Increases the ability of any pair to work on any story in the backlog
Sustainable Software Development

3 Principles
3 Policies
3 Practices for Removing Knowledge Silos
3 Practices for Caretaking the Code

ESEM 2016
Policies

Team Code Ownership
Increasing the ability to change any of the team’s code
“I feel ownership of the code as a whole.
I feel empowered to work on any part of the codebase.”

Shared Schedule
Working the same hours whenever possible

Avoid Technical Debt
“Thinking that someone else will fix it can be dangerous,
because then nobody will do it.”
Removing Knowledge Silos Practices

**Continuous Pair Programming**
Pairing whenever possible
Increases Team Code Ownership

**Overlapping Pair Rotation**
Rotating one partner on each track of work
“This reduces knowledge silos and reduces the bus factor. We do not want the departure of one developer to cripple the project.”

**Knowledge Pollination**
Sharing information in unstructured ways
Example: Removing Knowledge Silos
Individual Code Ownership

A ➔ B ➔ C ➔ D ➔ E ➔ F

↔ Knowledge Transfer

Knowledge Gap
Continuous Pair Programming

A ↔ B  C ↔ D  E ↔ F

↔ Knowledge Transfer

Knowledge Gap
Overlapping Pair Rotation

Strategies
- Optimizing for people rotation
- Optimizing for personal preferences
- Optimizing for context sharing

Knowledge Transfer
Knowledge Gap
Knowledge Pollination

A        B

F        E

D        C

↔ Knowledge Transfer

↔ Knowledge Gap
Knowledge Pollination

Contributing to knowledge sharing in an un-structured way
Daily stand-up meetings
Weekly retrospectives
Writing or sketching on whiteboards
Overhearing a conversation
Using the backlog to communicate current status about a story
Calling out an update to the entire team
Simply reaching out to team to ask a question
Example:
Strategies for Rotation
Strategies for Rotation

- Adhoc
- Random
- Optimized for Personal Preference
- Optimized for People Rotation
- Optimized for Knowledge Sharing
Four Tracks of Work

Track 1

Track 2

Track 3

Track 4

Icons designed by Freepik
Individual Ownership

Day 1  Alice  Bob  Generated Knowledge After Four Days
Day 2  Alice  Bob
Day 3  Alice  Bob
Day 4  Alice  Bob
Optimizing for Knowledge Sharing

Day 1
Alice
Bob
Day 2
Carol
Day 3
Dan
Day 4

Has enough context been shared?

Day 1: Alice - a, Bob - a
Day 2: Carol - b, Bob - b
Day 3: Dan - c, Carol - c
Day 4: Dan - d, Carol - d

Has enough context been shared for four days?
Caretaking the Code Practices

TDD / BDD
Write unit tests before creating a design or writing code
Creates a safety net
“Allows me to change code without breaking everything”

Continuous Refactoring
Systematic improvement of the code base
Enables any pair to work on any part of the system

Supported by Live on Master
“Use frequent commits and always rebase”
Conclusion
Summary

1. Introduces Sustainable Software Development Theory
Summary

2. The theory extends our understanding of Extreme Programming

3. Mitigates the negative effects of major disruptions by
   i. Spreading knowledge around the team
   ii. Writing “well-factored” code
Conclusion

Primary benefits to the software developer
Understand the team’s code
Work on any story
Increased teaching opportunities

Primary benefit to the employer is business agility
Team continues to deliver “quality software effectively” while surviving disruptions
Thank You!
Sustainable Software Development paper

Available at ResearchGate.com
Discussion
At Pivotal

Anchors can use the paper to push back on Technical Debt

Anchors can be intentional about removing knowledge silos
  • I recommend "optimizing for knowledge sharing"

Teams can strive towards the ability for any pair to work on any story in the backlog
Dimensions of Software Engineering Success

1. Impact on Stakeholders
2. Project Efficiency
3. Artifact Quality
4. Market Performance
5. (The Other) Time

Ralph 2014
It's a simple linear function: the more coders we use, the less time it takes!

No no, it must be exponential, as we can't keep adding more coders forever!

No no no, it's even worse than that. It's a hyperbolic curve!

Whatever, it doesn't matter. The domain of the function is \( \{1\} \)

No way I'm letting you touch my code....
Discussion