Pair Programming

CS312
Spring 2012

Slides adapted from:
Bob Kessler and Laurie Williams

Admin

- Assignment 4 scores sent out
- Assignment 5 soon…
- Tech talk schedule set
  - If you need to change for some reason
    - negotiate with others in the class
    - send me an e-mail with the update

Admin

- Project
  - Make sure to keep track of your sprint updates
  - Git
    - use branches for major features
    - figure out practices as a team what work best
    - communication is crucial!
    - http://kentnguyen.com/development/visualized-git-practices-for-team/

Admin

- Project
  - Pep talk ☺
    - this is new for everyone
    - it's going to take a while to figure out where the project is going
    - working in a new team is challenging
    - rails is new for most of you
    - I understand all of this…
      - don't get frustrated
      - just keep putting in the time
      - you'll get there… baby steps!

What is pair programming?

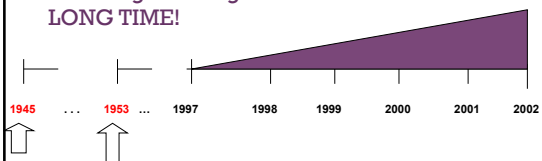
- Many variations
- Basic ideas:
  - two people working on one piece of code
  - continual communication between the two programmers
  - only one person is editing the code at any time

Pair programming

- Driver:
  - person currently working on the code
- Observer:
  - watches the person coding
  - looks for errors
  - should be thinking higher-level about where to go next, issues, etc.
- Switch rolls frequently and regularly (e.g. every 5-10 min)

This Is Pair Programming

Pair Programming Has Been Around For a LONG TIME!



John von Neumann, recognized his own inadequacies and continuously asked others to review his work.

Fred Brooks and many others are pair programming, though they don't know there is a name for it.

Pair programming

Benefits?

Drawbacks?

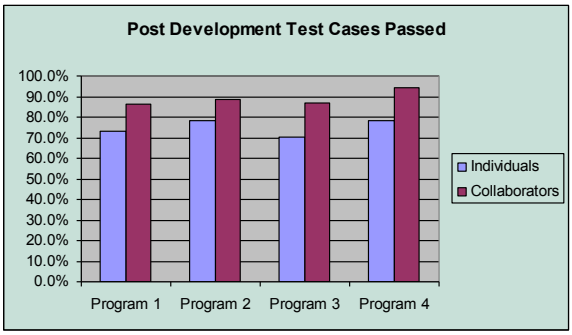
Does Pair Programming Really Work?

It depends…

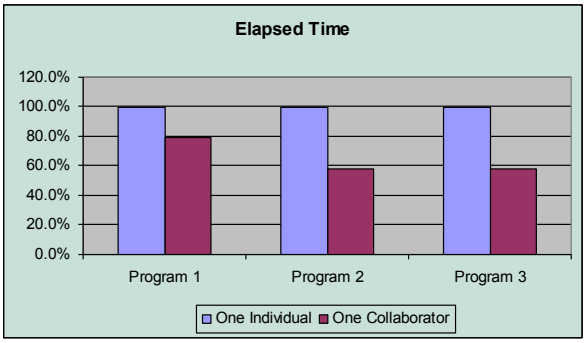
Does Pair Programming Really Work?

- Many different studies (Wikipedia cites some, but others exist)
- Empirical study by Laurie Williams at the University of Utah (now at UNC)
  - Solo vs. pair: Fall 1999
    - 41 students (junior/senior)
      - 28 worked collaboratively
    - 13 worked individually
    - Software development process was controlled
      - The only experimental variable: pair-programming
    - Quantitative: time, quality, enjoyment, confidence

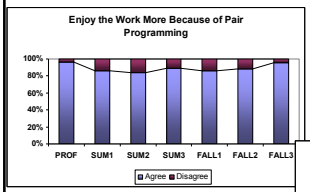
Findings #1 - Quality

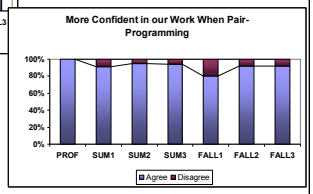


Findings #2 - Time



Findings #3 and #4 – Enjoyment and Confidence





Why does it work?

- Pair-Pressure
  - Keep each other on task and focused
  - Don't want to let partner down
  - "Embarrassed" not to follow the prescribed process
  - Parkinson's law: *"work expands to fill all available time."*

Why does it work?

- Pair-Think
  - Distributed cognition: "searching through larger spaces of alternatives"
    - Have shared goals and plans
    - Bring different prior experiences to the task
    - Different access to task relevant information
    - Must negotiate a common shared of action

Why does it work?

- Pair-Relaying
  - Each, in turn, contributes to the best of their knowledge and ability
  - Then, sit back and think while their partner fights on

Other Research Findings

- Strong anecdotal evidence from industry
  - "We can produce near defect-free code in less than half the time."
- Empirical study
  - Pairs produced higher quality code
    - 15% less defects (difference statistically significant)
    - Observed – pairs produced smaller (LOC) programs
  - Pairs completed their tasks in about half the time
    - 58% of elapsed time (difference NOT statistically significant)
  - Most programmers reluctantly embark on pair programming
    - Pairs enjoy their work more (92%)
    - Pairs feel more confident in their work products (96%)

Other Research Inquiries

- Other questions:
  - What about pair learning?
    - Anecdotally people say that it works well
  - What are the long-term issues?
    - If you learn as a pair, can you work as a solo?
  - Can pair programming be done remotely?

Issues: Partner Picking Principles

How do you pick pairs?

What are some things to watch out for?

Issues: Partner Picking Principles

Expert paired with an Expert

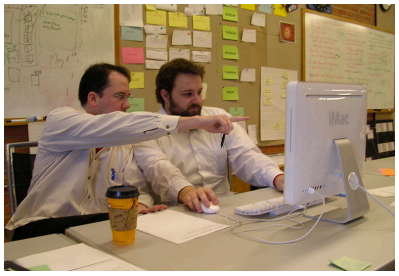
Expert paired with a Novice

Novices paired together

Professional Driver Problem

Culture

Configurations



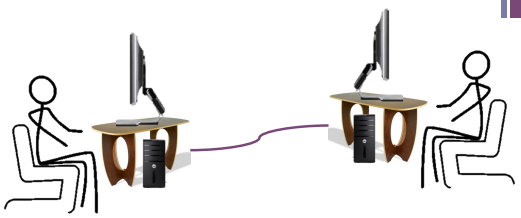
Most common: sitting next to each other

Configurations

Face-to-face with own screen and own keyboard/mouse but using a shared editor

- allows for face-to-face conversation
- some feel more natural and productive

Configurations



Remotely using a shared editor

Expected Benefits of Pair Programming

- Higher product quality
- Improved cycle time
- Enhanced learning
- Pair rotation
  - Ease staff training and transition
  - Knowledge management/reduced product risk
  - Enhanced team building
- Increased programmer satisfaction

Some concerns

- There are some barriers to entry
  - office configurations
  - people are resistant
  - holds people accountable
  - http://blog.obiefernandez.com/content/2009/09/10-reasons-pair-programming-is-not-for-the-masses.html
- not faster in all situations
  - you are devoting two people to a task
  - some tasks are very straightforward and may only require one person

Pair programming applied

- Implement binary search in ruby
  - two parameters:
    - a single sorted list
    - a value to search for in that list
  - returns
    - nil if it does not occur in the list
    - the index if it does occur (any index if it occurs multiple times)
- Pair program
  - switch every 3 minutes
- Two approaches
  - recursively:
    - use a helper function
    - pass a start and end index where you are still searching
  - iterative:
    - keep track of the start and end index where you are still search