Pair Programming, Pair Teaching - and beyond

Lars Bendix
bendix@cs.lth.se

Department of Computer Science
Lund Institute of Technology
Sweden

Parma, Italy, May 18th, 2006
Pair Programming

Pair programming is two people programming together at a single computer.
Pair Programming

Two programmers working side-by-side collaborating on the same design, algorithm, code or test. One programmer, the driver, has control of the keyboard/mouse and actively implements the program. The other programmer, the [navigator], continuously observes the work of the driver to identify the tactical defects and also thinks strategically about the direction of the work. On demand, the two programmers periodically switch roles, they work together as equals to develop software.
Cost-effective (soft)?

Two people doing one person’s job?

According to Kent Beck - and others:

- gives automatic code review
- two brains work better than one
- four eyeballs see more than two
- (most) people program faster in pairs
- design and code has higher quality
Cost-effective (hard)!

100% increase in costs?
No, a 60% increase in cost - but a reduction in time!
Going down to a 15% increase after a few tries!

Better code?
Single: 70-80% “quality” - Pairs: 85-95% “quality”

Do they like it?
Satisfaction and confidence > 90%
Experience

• you work more focused (avoid false trails)
• you work more effective (no “surf-breaks”)
• you get tired!!!
• better code quality (better detailed design)
• you stick to the method (TDD, refactoring, ...)
• you learn from your partner (design, tools, ...)
• easier to integrate new people
• you don’t get interrupted by other people
• many are negative before - but positively surprised
Pair teaching is two people teaching together in the same room.
Pair Teaching

Two programmers working side-by-side collaborating on the same design, algorithm, code or test. One programmer, the driver, has control of the keyboard/mouse and actively implements the program. The other programmer, the navigator, continuously observes the work of the driver to identify the tactical defects and also thinks strategically about the direction of the work. On demand, the two programmers periodically switch roles, they work together as equals to develop software.
The eXtreme Teaching Framework

Product: Student Learning

Practices
- Collective Course Ownership
- Explicit Rules of The Game
- Pair Teaching
- Student Involvement
- Constructive Alignment
- Planning Game
- Formative Assessment
- 40-hour Week

Feedback Communication Respect Courage

Iterative Scholarly Approach
Influences

Own experience:
  • craft’s apprenticeship
  • peer learning

eXtreme Programming:
  • Pair Programming
  • Team effort

Pair vs. team/group/collaborative teaching
Communication

Moving/spreading/sharing information:
  • past/present/future
  • costs/barriers/overhead

Lunchroom/corridor chatting
Courage

Someone with whom to brainstorm (and debug) ideas.

Someone to help you out of troubles.
Respect

People have different ways of thinking.

Dialogue increases the understanding.
Feedback

Data collection.
Pair Teaching

Solo workers are more likely to make mistakes

Preferable from a scholarly approach:
  – Someone to share (develop) ideas with
  – Puts your ideas to peer review
  – Developing colleagues

More occasions for reflection/improvement:
  – Partner “forces” you through Kolb’s learning cycle

Back up flexibility

Values:
  communication, courage, respect, (feedback)
Example: Running a course together

All major activities are done together except for the lectures (though pair planned)

Especially two outcomes stand out:
• The quality assurance aspect – all ideas must be made explicit to the other, then they are directly peer reviewed
• A bonus back up flexibility – things can be carried out alone even if planned for pairs
Example: Developing a new course

Course outcome and learning goals.
  • done together - experienced driver, senior navigator

Structure and contents of lectures.
  • done together - equal driver/navigator

Preparing and reviewing lectures.
  • prepared alone - peer reviewed

Attending/supervising lectures.
  • red thread - feedback/data

Exercises and computer labs.
  • done together - one “asking”, the other “answering”
Example: Student Pair Teaching

Coaching in pairs.

Students doing Pair Programming:
  • gap decreases
  • strong learns from teaching
  • weak learns from being taught (hands on)
  • best learning for both if strongest is navigator

Coach-student pairing.
Example: Passing on a course

Getting into the “philosophy” of the course.

Guessing at thee “motivation” for the material:
  • slides
  • exercises
  • labs
  • assignments

Discovering that some things were too complex - and others were too easy.
Cost-benefit

Two people doing one person’s job?

According to Kent Beck - and others:

• gives automatic code review
• two brains work better than one
• four eyeballs see more than two
• (most) people program faster in pairs
• design and code has higher quality
Cost-benefit (soft)

Immediate and explicit benefits:
• Having someone to brainstorm with
• Having someone to “debug” your ideas
• Having a possibility to handle larger groups
• Having the ability to step in for each other

More long-term or “hidden” benefits:
• Communicating information
• Assuring quality aspects
• Educating colleagues
• Guide the faculty towards SoTL

Done the right way, the benefits outweigh additional costs!
Pair studying could be two people studying together at the same table.
Pair Studying

Two programmers working side-by-side collaborating on the same design, algorithm, code or test. One programmer, the driver, has control of the keyboard/mouse and actively implements the program. The other programmer, the [navigator], continuously observes the work of the driver to identify the tactical defects and also thinks strategically about the direction of the work. On demand, the two programmers periodically switch roles, they work together as equals to develop software.
Pair Studying?

• reading the text (book/papers/notes/...)
• attending a lecture (before/during/after)
• doing exercises (before/during/after)
• doing labs (before/during/after)
• preparing exams (before/during?/after)
• making a thesis together???
• ....
Cost-effective?

Pair Programming used in a web programming class:
- 74% “between me and my partner, we could figure out everything”
- 84% “I learned ASP faster and better because I was always working with a partner”