Pair Programming

What’s in it for me?

Paper by A. Begel and N. Nagappan, Microsoft Research
published in ESEM 2008

Presentation by Martin Senn,
Software Engineering Seminar 2010
March 7th, 2010
**Definition:** Practice in which **two** programmers work collaboratively at **one** computer on the same design, algorithm or test.
Pair Programming

**Driver**

actively types at computer

writes code, UML, design documents, etc.

**Navigator**

watches the work of the driver

identifies defects, errors, makes suggestions

The two collaborate, discuss and brainstorm
Pair programming wastes two programmers to do the job of one!
Pair programming results in fewer bugs and better quality code.
Pair Programming \textit{in theory}

- part of the Extreme Programming Methodology

- should give the following benefits:

1. \textbf{Better quality:} many mistakes/bugs are found early in the development process (as they are being typed)

2. \textbf{Reduced cost:} bugs are very expensive and as they can be found early, the development cost is reduced

- early research results indicate:

    The code and design quality is better, coding needs more time, but only about 15\%, but the cost for testing and quality assurance is reduced.

    “The Costs and Benefits of Pair Programming”, Williams et. al., 2001
- a friend of mine who practiced pair programming told me:

1. ) “In the pair we produced less code and less bugs. The code quality was really better! We not only found bugs early but also could identify wrong approaches and designs early”

2. ) “The risk of getting lost is much smaller, as each programmer has a different perspective”

4. ) **But:** “Both programmers should have the same skill level e.g. regarding design patterns”

5. ) **Important:** “The programmers should reverse the roles of driver and navigator”
Pair Programming *in academic setting*

- Previous research on pair programming mostly in this area
- Surveys of students of different universities

**Perceived Benefits:**
- students can learn from each other
- students can develop inter-personal skills

**Findings:**
- pairing should be done between students of similar or equal skill level!
- pair programming students produce better quality code and better designs
- Time consumption ranges from half the time to twice the time of a single programmer!
Pair Programming *in industrial setting*

- Only little research on pair programming in industry
- What previous results indicate:
  - learning is not the primary goal (as in the academic setting)

Previous Findings
- pair programming developers *may not necessarily* produce better quality code!
- The effort to complete tasks correctly is in some cases *reduced*, sometimes *increased*!
The Paper:

- Paper based on asking developers directly
- 487 survey participants, from around the world

  21% of survey respondents have practiced pair programming (only) in the past
  3.5% practice it in current projects

- Only 106 participants had pair programming experience (these are the ones we’re interested in)

  68% developers
  21% testers
  8% managers
  Rest: researchers, etc.
Survey:

- Software engineers had to answer questions about:
  - perceived **benefits** and **problems**
  - **characteristics** of a good pair programming partner and team
  - influence on **quality** and **productivity**

- First result:
Survey results

Perceived Benefits

1. **Reducing bugs:** bugs are found and fixed earlier 66%

2. **Higher quality code:** improved software quality because of constant code review and combined knowledge and skills 48%

3. **Spread code understanding:** spread knowledge about design and code across the team 42%

4. **Learn from partner:** each partner has a slightly different skill set and knowledge 42%

5. **Better design:** different opinions help to see problems from different viewpoints and to think about various approaches and alternatives 30%
Survey results

Perceived Problems

1.) **Cost:** people are being paid to do the job of one, needing twice as many people!  

2.) **Scheduling:** the two partners need the same, overlapping schedule  

3.) **Clash of personality:** project may suffer from bad pairing, as performance depends on compatibility  

4.) **Disagreement:** it’s hard to find a consensus in ideas, wasting times in discussions  

5.) **Skill differences:** people are worried that they are paired with a partner who is not as smart
Survey results

Characteristics of a good partner

1. ) **Complementary skillset:** your partner should have different ideas, opinions and a different background, the skillset should be overlapping but not identical!

2. ) **Flexibility:** your partner is “open-minded”, open to new ideas and can adapt to different styles

3. ) **Good communication skills:** your partner is a good listener, should have *good inter-personal skills*
Survey results

Characteristics of a good team

1.) Good communication skills: a team needs compatible communication styles

2.) Complementary skills: partners should have complementary skills and knowledge

3.) Compatible personalities: partners are not competing, discuss, share ideas and are tolerant

4.) Efficiency: partners' minds work similar and hence they don’t argue too much
Survey results

Main results I

“Pair programming wastes two programmers to do the job of one!”
Main results II

"Pair programming results in fewer bugs and better quality code."

Survey results
Pair Programming

Summary of findings:

- what was confirmed:
  - Primary perceived benefit: higher quality code
  - Primary perceived problem: higher cost

- what we learned:
  - Most programmers want to be paired with someone with complementary skills, who is flexible and has a compatible personality!
    - contrast to academic setting!
  - Programmers want to learn from each other
    - same as in academic setting!
Conclusion:

1 Good news: Pair programming can have the mentioned benefits!

2 Bad news: Also in order to be (cost) efficient the team and individual partners must have the mentioned qualities!

3 “Ray of hope”: If people are encouraged to work together, they can learn to work together and become an efficient team!