Table of Contents

What is Pair Programming?

> Or “what are the benefits of pair programming?”
> Or “how should I get started with pair programming?”

I've recently has a number of discussions about the benefits of pair programming. Some of the thinking was that the main use of pair programming was as a learning tool, to help break down our specializations. This is the result of the emphasis we made in answering the question of breaking down specialization on a team (see How Do We Get All the Work Addressed When Our Specialists Cannot Be Everywhere?) In fact, pair programming was first proposed as a tool to help developers do better work – fewer defects, better designs – and it was found that the practice also helps with this cross-learning.

First a definition. Pair programming is “All code to be sent into production is created by two people working together at a single computer.” James Shore has an article on Pair Programming at http://www.jamesshore.com/Agile-Book/pair_programming.html (this is excerpted from the Art of Agile Development) which explains the practice.

Perhaps the canonical “benefits” paper on this practice is “The Costs and Benefits of Pair Programming” by Alastair Cockburn (one of the guys who developed the Agile Manifesto) and Laurie Williams. They tried to tease apart, through interview and data, what pair programming was all about and the paper was their findings.

Benefits

In general people see the following benefits of using pair programming:

- Improved code quality as you catch defects earlier - it's a continuous code review.
- Improved code design as you are explaining decisions as you go and there is someone there to offer up suggestions
- Improved flow of work as the pair means that fewer things will block (or interrupt) the creation of code.
- Lower development costs as you find / fix defects when it's still cheap to address.
- Lower cost, as there is less of the find / fix cycle needed to support the code.
- Pair learning and no single point of failure as a result of increased number of people that become aware of the whole system.
- Team building as, let's face it, this is a collaborative effort.

Skeptisism

Most developers I have worked are skeptical of the idea somehow this produces better results. The thinking is that if you have 2 people doing the work where 1 used to do, it must be less productive. The

reality is that if development were just a matter of typing, then yes, having two people do the work of one does not make economic sense. But if you look at the way you develop, a lot of the time is actually spent thinking (this is knowledge work after all) and so there is expected benefit of working together to improve a result. In fact, you could take this idea to the extreme. There is a concept out there called “mob programming” one keyboard / mouse for the whole team, with one team member driving and everyone else navigating (see https://www.youtube.com/watch?v=p_pvslS4gEI if you don’t believe me – and no I am not advocating this as I haven’t actually tried it myself).

Bottom line concept:

> “Pair programming is pair 'thinking' not pair 'typing'!”

Experiment

There is also significant reticence in adopting the practice as it is so different from current “solo” development practice. So my recommendation is to “try it”. Take a story that you are working and try to do it. See if there is benefit other than shared knowledge.

Want To Know More?

General view:

- http://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF - Alastair Cockburn. Pretty much the study that everyone quotes from. From the summary: “The significant benefits of pair programming are that · many mistakes get caught as they are being typed in rather than in QA test or in the field (continuous code reviews); · the end defect content is statistically lower (continuous code reviews); · the designs are better and code length shorter (ongoing brainstorming and pair relaying); · the team solves problems faster (pair relaying); · the people learn significantly more, about the system and about software development (line of-sight learning); · the project ends up with multiple people understanding each piece of the system; · the people learn to work together and talk more often together, giving better information flow and team dynamics; · people enjoy their work more. The development cost for these benefits is not the 100% that might be expected, but is approximately 15%. This is repaid in shorter and less expensive testing, quality assurance, and field support.”
- http://www.extremeprogramming.org/rules/pair.html - Extreme Programming (.org). Definition “All code to be sent into production is created by two people working together at a single computer. Pair programming increases software quality without impacting time to deliver. It is counter intuitive, but 2 people working at a single computer will add as much functionality as two working separately except that it will be much higher in quality. With increased quality comes big savings later in the project.”
- http://www.jamesshore.com/Agile-Book/pair_programming.html - James Shore on Pair Programming. Excerpted from the Art of Agile Development
- https://www.researchgate.net/publication/3249409_Are_Two_Heads_Better_than_One_On_the_Effec

tiveness_of_Pair_Programming - Jo Hanney (et al). Includes discussion about relative effect of application such as complexity of the problem.
- https://pragprog.com/magazines/2011-07/pair-programming-benefits - List of “developer” and “management” benefits to pair programming
- https://scholar.google.com/scholar?q=pair+programming+study&hl=en&safe=off&esrch=BetaShortcuts&um=1&ie=UTF-8&oi=scholart – scholarly search for research on pair programming.
- https://en.wikipedia.org/wiki/Pair_programming - Wikipedia view.

Note that not everyone has a positive view of the practice:

- http://namcookanalytics.com/high-costs-and-negative-value-of-pair-programming/ - Capers Jones doesn’t think it works at all for large scale development (10,000 people) based on lines of code cost and some analysis of function points. Assumes that the only way to deliver “big” is to deliver serial. Comparison seems to say “you can produce same results of defect reduction with code inspection (a group of people looking at the code?)” but does not include this cost in the development cost or at least not in a way I could find.

FAQ, benefits, XP, PairProgramming