> We believe that Pair Programming allows us to better serve our customers' needs and affords them the best value for their investment

PAIR PROGRAMMING

WHAT IT MEANS TO ASYNCHRONY...WHAT IT MEANS TO OUR CUSTOMERS.

VERY EARLY IN ASYNCHRONY'S HISTORY, THE COMPANY EMBRACED AGILE METHODOLOGY AND EXTREME PROGRAMMING, INCLUDING THE CONCEPT OF PAIR PROGRAMMING. JUST AS WE ARE COMMITTED TO THE BROAD BENEFITS OF AN AGILE DEVELOPMENT METHODOLOGY, WE ALSO BELIEVE THAT PAIR PROGRAMMING ALLOWS US TO BETTER SERVE OUR CUSTOMERS' NEEDS AND AFFORDS THEM THE BEST VALUE FOR THEIR INVESTMENT.

Numerous studies have shown that Pair Programming improves productivity and design quality with minimal economic impact to the customer. In fact, some evidence shows that customers recoup any short term cost increases with long term cost savings due to the quality of code produced in Pair Programming environments. Pair Programming is also conducive to higher innovation and better problem solving, as programming pairs tend to talk through issues more thoroughly and catch more errors during the coding process. Utilizing rotating paired development teams not only improves quality, but also significantly reduces the risk to the project associated with turn-over or developers not being available at key times due to sickness, vacation or other reasons. The result is better code, which in turn results in shorter testing cycles.

There are many broader indirect benefits of Pair Programming for a development organization. Our employees report high job satisfaction and a sense of open, team-oriented communication. They also enjoy learning from each other and developing cross-functional knowledge. Asynchrony places high value on employee satisfaction, and equally trusts our employees' commitment to customer satisfaction. The impact on a team's productivity due to low turn-over is a critical direct benefit to our customers derived from highly satisfied employees.

Our own beliefs about the benefits of Pair Programming are supported by the research of respected professionals in our field.

THE COSTS AND BENEFITS OF PAIR PROGRAMMING

Alistair Cockburn, Humans and Technology and Laurie Williams, University of Utah Computer Science

In Alistair Cockburn's and Laurie Williams' paper on the Costs and Benefits of Pair Programming (Attached as Exhibit I), the authors "examined interview and experimental data to understand the costs and benefits of" Pair Programming. In the paper, they both reference other studies that have been conducted that show the tangible benefits of Pair Programming, as well as extend the existing research in this area. Their conclusions are as follows:

"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.

Through Asynchrony's relationship with Gartner, we have access to research by top-rated analysts and can share these findings with our customers. Below are excerpts from two articles published by Gartner, followed by an anecdotal post by a third party developer that addresses many of our general assertions about Pair Programming.

DRIVING QUALITY UPSTREAM: EFFECTIVE CODE REVIEW PRACTICES

25 June 2008
Gartner Research Analyst: Thomas E. Murphy
ID: G00158218

Excerpt:

PAIR PROGRAMMING

One of the practices of extreme programming (see "Agile Essence: Extreme Programming") is the concept of pair programming. Two developers work together at the same machine:

One developer is in charge of the keyboard and the other is in charge of the design; thus, the developers explain their design to each other and get instantaneous feedback, rather than just entering code or communicating electronically. There are many variations of this model, including pairing a developer and a tester, having pairing sessions interspersed with coding sessions, and pairing new and experienced developers for knowledge transfer and training. This style of development can be shocking to developers who often are accustomed to working in isolation. *Organizations that have been successful with pair programming have seen strong gains in productivity and application quality.*

IS LEAN AN AGILE DEVELOPMENT METHOD?

20 May 2008
By Gartner Research Analyst: David Norton
ID: G00157544

Excerpt:

KEY FINDINGS

- Lean software development is a strategic, continuous improvement process for development, not a short-term tactical approach.
- Lean and agile are complementary and should not be seen as mutually exclusive.
- Lean and agile continue to coalesce around a common set of principles and practices.

Recommendations

- Organizations that want to improve waterfall development processes will find value in lean. However, at a certain point, it makes sense to adopt a more-standard, agile method, such as Extreme Programming (XP).
- Development teams looking to move to agility will find adopting XP, Scrum or Dynamic Sys-tems Development Method (DSDM) more successful in the short term compared with lean.
- Business and IT organizations will find a blended approach more successful — mainstream agile (for example, XP, Scrum and DSDM) in the context of a lean improvement framework.

SCOPE IN THE IT ORGANIZATION

Agile principles and practices were born out of the failings of traditional prescriptive methods, such as waterfall. The drive for shorter cycle times, improved planning and estimation, and the need to focus on business value have given rise to Scrum, XP and many other methods. Based on their common heritage, they have a common scope — that is, developers and development. *The XP practices of pair programming, integration and collective code ownership anchor XP at the project level.* Scrum focuses on management of projects using agile principles of a short cycle time, business feedback and empirical control, but still in support of development.

WHY PAIR PROGRAMMING?

27 May 2009
By: Geir Berset
http://aptoma.com/select.star/2009/05/27/why-pair-programming/

I am a sucker for rationale. I've been struggling with rationale on the Extreme Programming (XP)-practice of pair programming for quite some time. What at first looks like one person writing code, and the other one watching, has admittedly been very counter intuitive to me. Let me share my current thoughts of benefits and disadvantages of so called pairing.

Disadvantages of Pair Programming: Don't four hands produce more code than two?

If you are a carpenter, you can hit more nails with four hands than with two. If you are a brick-layer you can lay more bricks with four hands than with two. If you are a developer you can write more production code with four hands than with two. Or wait. This is based on the assumption that the bottleneck in software development is a mechanical one; the speed of which you can type on the keyboard. My 10 years as a developer and 5 years as a program manager tell me otherwise. The bottlenecks in somewhat innovative projects (non-repetitive) rarely or never include typing speed. Bottlenecks do include hesitation (doubting your own solution), introducing bugs and having to go back to fix it, suboptimal design that deserves a refactoring, lack of needed knowledge for the task, loss of focus and so on.

A small disclaimer before I go on: We have not been experimenting with Pair Programming for more than a few months, and my above observations are based on observing teams develop from a managerial point of view and from reflecting upon my own coding habits as a team member.

Benefits of Pair Programming: Amplify learning

The speed of learning in pair programming is overwhelming. Once you enter a topic one party has more experience in, you are in for a ride. I have never experienced learning at such a pace as when this occurs.

Prevent bugs

Bugs are cheapest when they are prevented from being introduced with measures that do not put restraints on your risk-taking and innovation. Pairing keeps you innovating, and will still help you spot errors more frequently. You will be more critical of your own code, and you will have someone constantly looking for possible improvements in your code, such as stopping a bug in the making, or having an idea for an even better test.

Continuous improvement

Pair programming is a dialogue between two people trying to simultaneously program, and understand how to program better.

Articulating ideas

Pair programming forces you to put words to your thoughts. This will through constant practice improve your skills in articulating your ideas.

Share frustrations

The responsibility on one single programmer can be huge at times. It is relieving to carry the load together. A team's strength and robustness is greater than the sum of strengths.

Avoid hesitation

When exhausted for ideas or new angles, it can lead to hesitation to implement what you suspect is a sub-optimal solution. The union of ideas and angles between the two developers makes this a rarer incident, and the resulting discussions will usually help remove the knots causing hesitation.

Distributed resources

From a manager's point of view, a migration of people between project modules, and even between projects, becomes vastly easier. This is a dream come true. Nobody is the sole "owner" of code anymore, and developers can move more freely between code sections and projects, simplifying the complex task of resource scheduling. Starting out on unfamiliar code means that you should be pair programming with someone with a lot of domain knowledge in the start.

How to do pair programming

Pair programming is two people using one keyboard to write code. Here are some important best-practices to go along with that description.

1. Choose a suitable partner for pair programming the given task during the daily scrum.
2. Sit side by side with only one keyboard and one mouse. Both should have a good view of the monitor.
3. It must be easy to slide the keyboard from person to person. Switch roles from typer to observer every so often.
4. Discuss and have a good time.

Pair Programming is a software development technique that supports Agile or Lean software development *methodologies*. These methodologies have proven their worth providing increased customer collaboration, customer satisfaction, and overall project success. While opinions may vary as to whether or not two developers working together affects the cost of a project, a *reduced* development cost is a reported benefit. *Wikipedia* states that, "with bugs being a particularly expensive part of software development, especially if they're caught late in the development process, the large reduction in defect rate due to pair programming can significantly reduce software development costs" (**http://en.wikipedia.org/wiki/Pair_programming**). Equally important from a customer perspective, Pair Programming reportedly decreases management risk and improves development flow. Metrics from scientific studies on Pair Programming that show gains in productivity are also reported in the above *Wikipedia* article.

SUMMARY

Asynchrony is an Agile software development firm that embraces the practice of Pair Programming. Our Agile methodology and our commitment to excellence require that we listen and respond to our customers on a continual basis. The information provided above is only intended to serve as reference material. We know that the best way we can address our customers' concerns is to engage in meaningful conversation leading to a shared understanding and a clear path forward.

Additional Reference attached:

The Costs and Benefits of Pair Programming, by Alistair Cockburn and Laurie Williams.
http://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF

ABOUT ASYNCHRONY LABS

Asynchrony Labs is an information technology consulting firm located in St. Louis, Missouri. We specialize in application development, mobile computing, systems and sensor integration, enterprise architecture, and tactical collaboration.

Our diverse client base includes commercial, non-profit, and government organizations. We've delivered solutions ranging from back-end government middleware to front-end applications and full-scale, commercial Cloud infrastructures. In short, Asynchrony Labs connects people, sensors, information, and systems.

Our mission is to create high-impact business solutions through ongoing client collaboration, iterative development, and continuous testing that are better than could have been envisioned at project start.

In 2015, Asynchrony joined World Wide Technology, an award-winning technology integrator and supply chain solutions provider that brings an innovative and proven approach to how organizations discover, evaluate, architect and implement advanced technology.

For more information, contact:
sales@asynchrony.com

900 Spruce Street, Suite 700
Saint Louis, MO 63102

314.678.2200