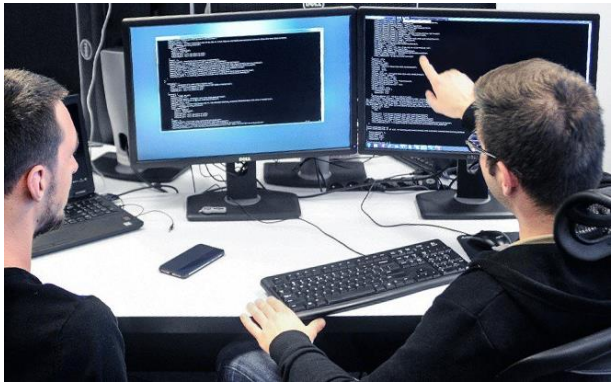
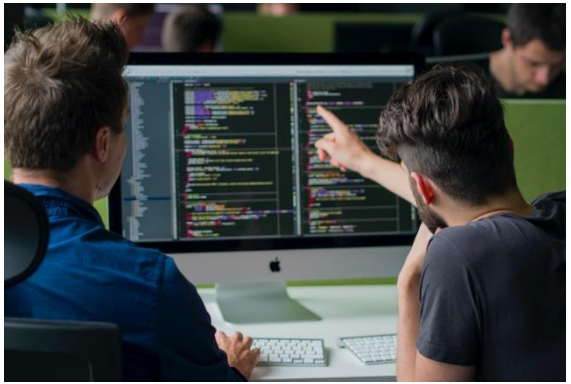
Linus Ververs
AG Software Engineering
Institut für Informatik
Fachbereich Mathematik & Informatik

Freie Universität Berlin

Pair Programming: What‘s in it for you?

A research perspective

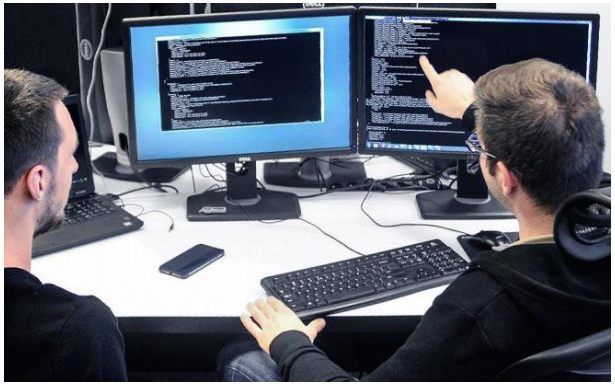
Pair Programming?

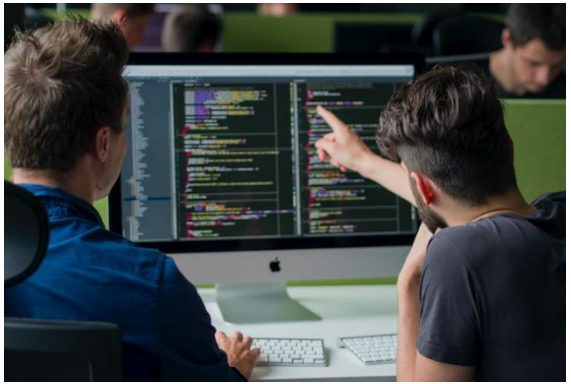




StackOverflow Developer Survey 2018 [1]

Which Methodologies do you use?

Pair Programming?





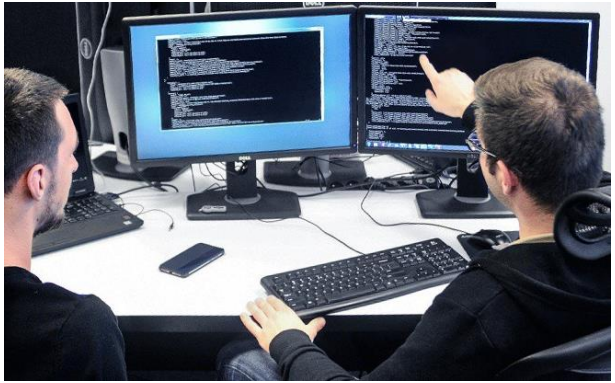
StackOverflow Developer Survey 2018 [1]

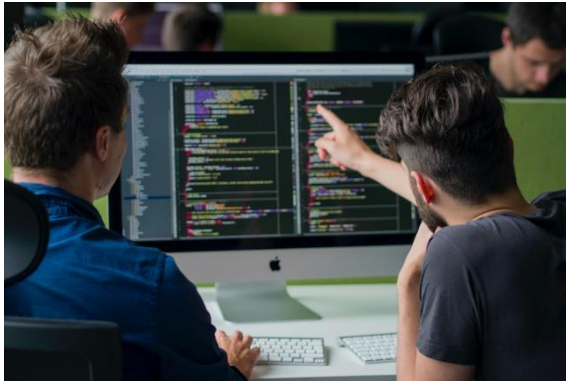
Which Methodologies do you use?

StackOverflow Developer Survey 2020 [2]

What do you do when you get stuck?

Pair Programming?





StackOverflow Developer Survey 2018 [1]

Which Methodologies do you use?

50%

Call a coworker or friend

StackOverflow Developer Survey 2020 [2]

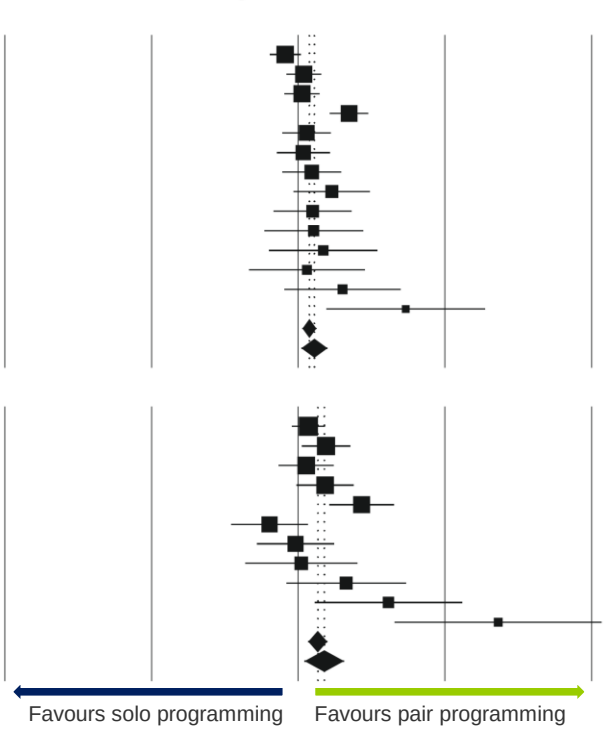
What do you do when you get stuck?

(Quasi-) Experiments on benefits of PP [3]

Most studies show that pair programming has a:

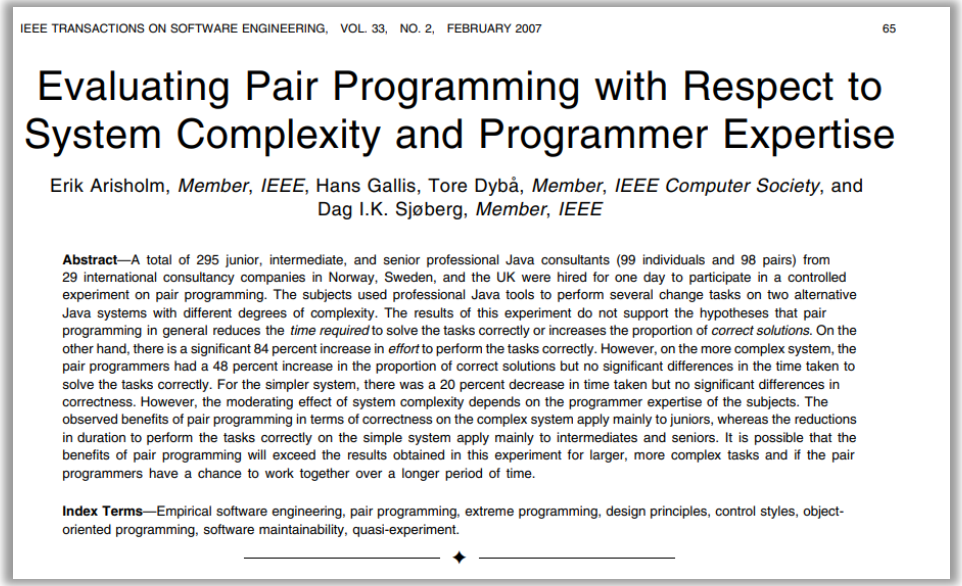
- Positive impact on quality.
- Positive impact on task duration.
- Negative impact on effort.

Study	g	95% C.I.		p
Quality				
Domino et al. (2007)	-.27	-.58	.05	.10
Arisholm et al. (2007)	.11	-.24	.46	.52
Madeyski (2006)	.08	-.28	.43	.67
Williams et al. (2000)	1.04	.65	1.43	.00
Madeyski (2007)	.17	-.32	.67	.49
Heiberg et al. (2003)	.10	-.44	.64	.71
Müller (2005)	.28	-.32	.88	.36
Canfora et al. (2007)	.69	-.09	1.46	.08
Baheti et al. (2002)	.30	-.50	1.09	.46
Phongpaibul & Boehm (2006)	.32	-.69	1.32	.54
Müller (2006)	.51	-.59	1.62	.36
Phongpaibul & Boehm (2007)	.18	-1.00	1.36	.77
Nosek (1998)	.91	-.28	2.10	.13
Xu & Rajlich (2006)	2.20	.58	3.82	.01
Overall fixed model	.23	.09	.37	.00
Overall random model	.33	.07	.60	.01
Duration				
Arisholm et al. (2007)	.21	-.13	.54	.22
Canfora et al. (2005)	.57	.07	1.07	.02
Nawrocki & Wojciechowski (2001)	.16	-.40	.73	.57
Heiberg et al. (2003)	.55	-.03	1.14	.06
Müller (2005)	1.30	.64	1.95	.00
Canfora et al. (2007)	-.59	-1.37	.20	.14
Baheti et al. (2002)	-.05	-.84	.74	.89
Rostaher & Hericko (2002)	.06	-1.08	1.21	.91
Müller (2006)	.98	-.24	2.20	.12
Xu & Rajlich (2006)	1.85	.34	3.35	.02
Nosek (1998)	4.09	1.98	6.20	.00
Overall fixed model	.40	.21	.59	.00
Overall random model	.53	.13	.94	.01



Pair Programming Research: One Example [4]

- Quasi-experimental study with 295 consultants
- Classification into Junior, Intermediate & Senior
- 99 solo developers vs. 98 pairs
- Processing of 2 tasks of different complexity



IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 33, NO. 2, FEBRUARY 2007 65

Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise

Erik Arisholm, *Member, IEEE*, Hans Gallis, Tore Dybå, *Member, IEEE Computer Society*, and Dag I.K. Sjøberg, *Member, IEEE*

Abstract—A total of 295 junior, intermediate, and senior professional Java consultants (99 individuals and 98 pairs) from 29 international consultancy companies in Norway, Sweden, and the UK were hired for one day to participate in a controlled experiment on pair programming. The subjects used professional Java tools to perform several change tasks on two alternative Java systems with different degrees of complexity. The results of this experiment do not support the hypotheses that pair programming in general reduces the *time required* to solve the tasks correctly or increases the proportion of *correct solutions*. On the other hand, there is a significant 84 percent increase in *effort* to perform the tasks correctly. However, on the more complex system, the pair programmers had a 48 percent increase in the proportion of correct solutions but no significant differences in the time taken to solve the tasks correctly. For the simpler system, there was a 20 percent decrease in time taken but no significant differences in correctness. However, the moderating effect of system complexity depends on the programmer expertise of the subjects. The observed benefits of pair programming in terms of correctness on the complex system apply mainly to juniors, whereas the reductions in duration to perform the tasks correctly on the simple system apply mainly to intermediates and seniors. It is possible that the benefits of pair programming will exceed the results obtained in this experiment for larger, more complex tasks and if the pair programmers have a chance to work together over a longer period of time.

Index Terms—Empirical software engineering, pair programming, extreme programming, design principles, control styles, object-oriented programming, software maintainability, quasi-experiment.

„However, on the **more complex system**, the **pair programmers had a 48 percent increase** in the proportion of **correct solutions** but no significant differences in the time taken to solve the tasks correctly.”[5]

Other notable Observations [4,5]

Duration of the task solving (minutes)

Scope			Unadjusted Means		
Prog. Cat	Term	CS	Ind	Pair	% diff
All	PP	CC+DC	87	63	-28 %
	PPxCS	CC	98	64	-35 %
		DC	72	62	-15 %
Junior	PP	CC+DC	92	81	-12 %
	PPxCS	CC	95	85	-10 %
		DC	86	78	-9 %
Intermed	PP	CC+DC	104	61	-41 %
	PPxCS	CC	113	60	-47 %
		DC	87	63	-27 %
Senior	PP	CC+DC	74	51	-31 %
	PPxCS	CC	87	53	-39 %
		DC	61	49	-20 %

Pairs are unaffected by complexity.

Quality of the task solving (Proportion of correct solutions)

Scope			Unadjusted Means		
Prog. Cat	Term	CS	Ind	Pair	Odds ratio
All	PP	CC+DC	60 %	82 %	3.01
	PPxCS	CC	68 %	80 %	1.93
		DC	51 %	83 %	4.68
Junior	PP	CC+DC	48 %	84 %	5.60
	PPxCS	CC	63 %	83 %	3.00
		DC	33 %	85 %	11.00
Intermed	PP	CC+DC	53 %	80 %	3.53
	PPxCS	CC	65 %	80 %	2.18
		DC	40 %	80 %	6.00
Senior	PP	CC+DC	75 %	82 %	1.48
	PPxCS	CC	76 %	79 %	1.15
		DC	74 %	84 %	1.90

With experience quality increases when working solo.

Constant (!) quality when working in pairs.

Experiments fail to capture reality

Tasks are small-scale.
There is no giant code base and no legacy code.

Partners do not possess system knowledge in advance of the session, that can be shared. Partners aren’t used working together.

Normative fallacy as root cause: “Pair programming is identical to solo programming and can be easily compared to it.”

Pair constellation and pair behaviour is crucial for success of PP.
There is good and bad PP. Not every task is suited for PP.

Our Approach: Qualitative field research

Observation / Analysis of software development done in practice.

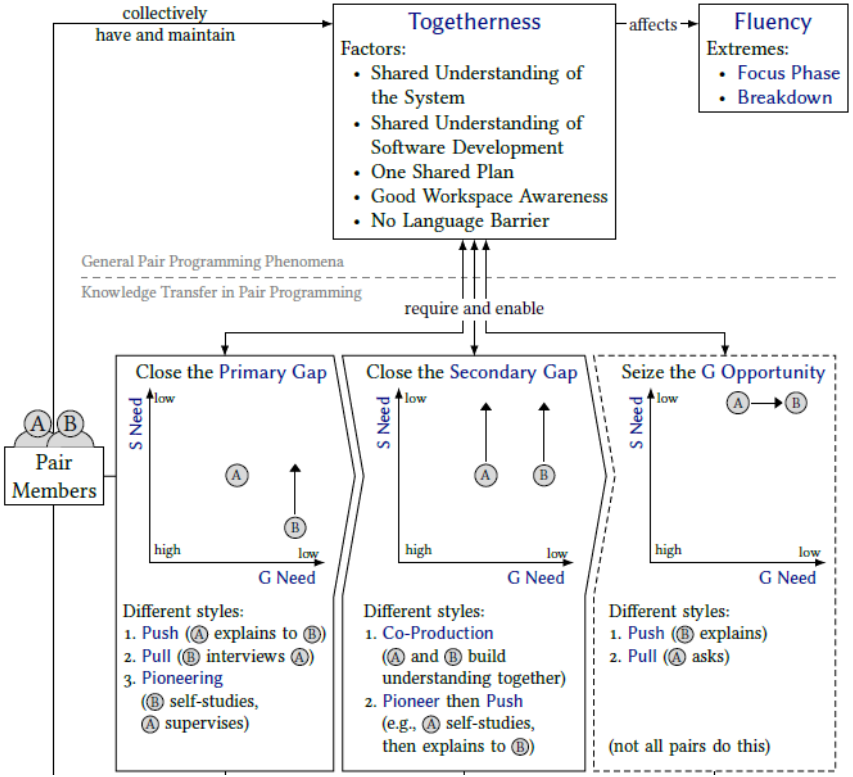
Analyse pair programming session as they happen naturally. Don‘t interfere with pair constellation, tasks chosen, duration, etc.

Identify, compare, and abstract positive and negative phenomena in each session. Formulate overarching theories.

Derive evidence-based advice and insights for improving software development in the industry.

Insights into knowledge transfer in PP [6]

- Togetherness as a fundamental success factor in PP.
- High togetherness leads to focus phases, low to breakdowns.
- Differentiation of S(system-specific)- & G(eneral)-Knowledge
- An equal level of task-specific S-knowledge is critical for togetherness.
- Knowledge transfer and acquisition in different styles.

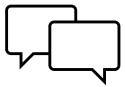


Discussion

Do you like or dislike pair programming? Why?

In your experience when did pair programming didn’t work at all? Why? What did you learn form it?

Do you practice pair programming on a regular basis? Why / Why not?



If you are interested in our research approach, please **feel free to contact me!** We are always looking for new companies and developers with whom we can collaborate.

Sources

[1] Stack Overflow Developer Survey 2018

[2] Stack Overflow Developer Survey 2020

[3] **Hannay, Jo E., et al.** "The effectiveness of pair programming: A meta-analysis." *Information and software technology* 51.7 (2009): 1110-1122.

[4] **Arisholm, Erik, et al.** "Evaluating pair programming with respect to system complexity and programmer expertise." *IEEE Transactions on Software Engineering* 33.2 (2007): 65-86.

[5] **Zieris, Franz.** The Normative Fallacy in Software Engineering Research.” Talk at Beiträge zum Software Engineering (2020).

[6] **Zieris, Franz. ”**Qualitative analysis of knowledge transfer in pair programming.” Diss. 2020.

Picture Sources

The Importance of Pair Programming for Developers | by Kayvan Kaseb | Software Development | Medium

Pair Programming Guide. Two heads are better than one —… | by Weblab Technology | Medium

Effective Remote Pair Programming (Part 1) (merits.com)