Pair Programming: Collocated Vs. Distributed

Mark Rajpal, M.Sc.
Agile Global Results
Calgary, Alberta, Canada
mark.rajpal@agileglobalresults.com

***Abstract*—Collocation is almost always a preferred alternative compared to Distributed. It makes sense that collocated team members are more likely to perform better than distributed team members. However, in today's real world the distributed nature is either the norm or quickly becoming the norm. That is not to say that collocation no longer exists, but rather it is becoming less and less pronounced. Pair programming is a technique that can be performed in a collocated or distributed fashion. Not all software development projects use this practice. The projects that do undertake this programming method typically perform collocated or distributed pair programming, but very rarely use both. This paper examines a project where both types of pair programming were used. At the completion of the project, all developers were asked to complete a survey. The results of the survey allowed us to compare various attributes of collocated and distributed pair programming. What may be surprising is that in some cases the differences between the two are minimal.**

Keywords—Agile; scrum; pair programming; extreme programming

I. INTRODUCTION

Pair programming is a concept that has become synonymous with Agile methodologies especially when it comes to extreme programming [1]. As two team members work together to develop software, one team member focuses on coding the task at hand while the other team member focuses on reviewing the code and thinking ahead to the next task. Team members may choose to swap these roles multiple times throughout the day.

The early days of pair programming suggested that collocated pairs should only apply the technique. However, as software projects become more dispersed, distributed pair programming has emerged as an alternative for distributed teams. This raises many questions. What are the constraints on distributed pair programming? Can two team members faced with temporal, cultural, and geographic barriers reasonably apply pair programming? Is it possible for distributed pair programming to achieve the same results (or better) than collocated pair programming?

A software development project (Project ABC) was undertaken for a power transmission company in the United States. The Scrum team was distributed throughout Canada where some team members were collocated. Upon completion of the project, the developers that participated in pair programming were asked to complete a survey based on their experience within this project.

The next section provides an overview of the project. Section III provides an overview of the various tools that were used. The progress of the project is covered in Section IV. Sections V and VI discuss the survey in detail. Additional Research and Conclusions are discussed in Sections VII and VIII.

II. PROJECT ABC

This project took place for the greater part of 2016. Some team members had little knowledge of Agile while others had no Agile experience whatsoever. However, the team was willing to try Agile as they were unhappy with previous phases that incorporated a modified Waterfall approach. The client also had no experience with Agile but was willing to try anything after multiple failed projects with a previous software vendor.

The single Scrum team consisted of four developers, one tester, and one requirements engineer. The Scrum methodology was used which included all Scrum ceremonies and each sprint was two weeks in duration. The author of this paper facilitated many roles which included Scrum Master, coach, facilitator, teacher, mentor, and team member.

Developers were encouraged to perform pair programming but were not mandated to do so. Reports have shown that pair programming is widely used technique across Agile teams (Table 1).

TABLE I. TEAMS USING PAIR PROGRAMMING

Report	Percentage
2016 State of Scrum Report [12]	35%
11th Annual State of Agile Survey [13]	32%

III. TOOLSET

The team utilized a multitude of tools to complete the user stories. Even though the team did not research specific pair programming tools, some of the chosen tools were a good fit for collocated and distributed pair programming.

A. Productivity

Since the team inherited a Java application, all developers agreed to use IntelliJ IDEA as their integrated development environment (IDE). There are many IDE options for Java and Eclipse was also considered but the team ultimately chose the commercial tool over the open source option. In hindsight, this decision may have restricted the team. IntelliJ does not provide many plugins that support distributed pair programming. While Eclipse has a few to choose from including, Sangam [2] and RIPPLE [3], other distributed pair programming tools that utilize their own editor including COLLECE and COPPER [4].

Various web based tools were incorporated that also enabled team members to conduct pair programming. These tools include; Jira, Confluence, Fisheye, Crucible, Bamboo, Nexus, and Bitbucket. All of these tools (except for Nexus) derive from the same software vendor Atlassian. This was a deliberate choice because tools from the same vendor generally integrate well together.

B. Communication

Everyone agreed that a good screen sharing tool was an absolute necessity. As a result, the team adopted TeamViewer. This allowed pairs to share their desktops and take over control when needed.

Regular phone calls were used when ad-hoc real-time voice communication was needed. However, most team members working out of their home location preferred Skype. There were many cases where Skype did not perform well. This could be attributed to low bandwidth associated with home Internet connections.

When voice communication was not required, pairs relied on HipChat to send instant messages (IM) back and forth. While HipChat does support video chat and screen sharing the team found it to be very problematic. Another IM tool called Slack was also considered but HipChat was preferred as the team had already incorporated other tools by that particular vendor (Atlassian).

GoToMeeting was utilized which also allowed for screen sharing. This tool was used in more of a pre-planned group setting. For example, sprint planning, sprint reviews, sprint retrospectives, and daily scrums were coordinated through GoToMeeting.

IV. Project Performance

The team estimated they could complete 50 story points worth of work in a 2 week sprint. Over the course of 15+ sprints the team exceeded that target every sprint. In some cases they overwhelming exceeded the target. The best sprint resulted in 476 story points.

Due to the high velocity, the original release plan was completed 3 months ahead of schedule. Instead of ending the project at that point the client requested additional work much to the satisfaction of all parties involved.

During the final 3 months of the project, the team was allowed to incur overtime. The Table 2 below shows the total hours for the entire project.

TABLE II. Pair Programming Hours

Pairing	Total Hours	Percentage
Collocated Pairs	2,794.5	47%
Distributed Pairs	3,149.25	53%

Table 2 shows an almost even split in terms of time invested. However, that does not indicate that both pairs were equally effective.

V. Survey

A. Overview

As mentioned, at the completion of the project, each developer was asked to complete a voluntary survey based on their pair programming experience. Since most of the developers had limited pair programming experience, their responses were directly related to Project ABC. The survey required participants to rate a series of questions from 1 to 10, where 1 represents “extremely disagree” and 10 represents “extremely agree”. There were some additional free form questions as well (Table 3).

TABLE III. Pair Programming Survey [Scale: 1 (Extremely Disagree) – 10 (Extremely Agree)]

#	Question	Combined	Collocated	Distributed
Q1	Pair Programming Enjoyment	M = 8.25	M = 8.5	M = 8
		SD = 0.5	SD = 0.71	SD = 0
Q2	Pair Programming Effectiveness	M = 8.5	M = 9	M = 8
		SD = 0.58	SD = 0	SD = 0
Q3	Pair Pressure (don’t want to let your pair down, sense of responsibility)	M = 5.25	M = 6	M = 4.5
		SD = 2.63	SD = 4.24	SD = 0.71
Q4	Pair Courage (did something you would not do if working alone)	M = 8	M = 9	M = 7
		SD = 2.83	SD = 1.41	SD = 4.24
Q5	Pair Programming increased productivity	M = 7.25	M = 8.5	M = 6
		SD = 1.71	SD = 0.71	SD = 1.41
Q6	Pair Programming increased quality	M = 8.25	M = 8	M = 8.5
		SD = 0.5	SD = 0	SD = 0.71
Q7	Pair Programming increased confidence	M = 7.5	M = 7.5	M = 7.5
		SD = 2.38	SD = 2.12	SD = 3.54
Q8	Pair Programming increased interest in Agile	M = 7	M = 5	M = 9
		SD = 4.08	SD = 5.66	SD = 1.41
Q9	Pair Programming reduced time spent	M = 7	M = 8	M = 6
		SD = 2.16	SD = 1.41	SD = 2.83
Q10	Were able to 'gel' with you primary pair partner	M = 7.75	M = 9	M = 6.5
		SD = 2.06	SD = 1.41	SD = 2.12
Q11	Prefer peer review to pair programming	M = 6	M = 6	M = 6
		SD = 2.94	SD = 4.24	SD = 2.83
Q12	Trustworthiness is important	M = 9.75	M = 10	M = 9.5
		SD = 0.5	SD = 0	SD = 0.71
Q13	Social etiquette is important	M = 9.25	M = 8.5	M = 10
		SD = 1.5	SD = 2.12	SD = 0
Q14	Energy is important	M = 7.75	M = 9	M = 6.5
		SD = 1.71	SD = 1.41	SD = 0.71
Q15	Expressing thoughts is important	M = 9.25	M = 9.5	M = 9
		SD = 0.96	SD = 0.71	SD = 1.41
Q16	Letting go of obsessions is important	M = 10	M = 10	M = 10
		SD = 0	SD = 0	SD = 0
Q17	Getting enough sleep is important	M = 8.25	M = 8	M = 8.5
		SD = 2.06	SD = 2.83	SD = 2.12

Each developer was asked to complete the survey individually. Furthermore, each developer (whether collocated or distributed) were given the exact same survey.

B. Results

The mean (M) and standard deviation (SD) was calculated for each question. Additionally, these results were represented amongst all developers, collocated developers, and distributed developers.

VI. Analysis

Overall, the results seem to indicate that pair programming was an effective tool for this project. Distributed and collocated pairs seemed to feel that Project ABC benefited from pair programming. In comparison, based on Q2, Q5, Q6, Q7, and Q9 collocated pairs seemed to recognize more benefits than their counterparts.

In terms of comradery (Q1, Q3, Q4, and Q10), collocated pairs reported higher findings than distributed pairs. This is not surprising given the fact that collocated pairs have daily face-to-face interaction. In fact, L. Williams [5] describes "pair fun" as an eighth behavior from the book, "Seven Synergic Behaviors of Pair Programming". Furthermore, after the completion of this project, many of the collocated pairs reported some sadness because they had moved onto other projects that did not require nor promote pair programming. This is consistent with the final stage of the Tuckman model 'adjourning' which is sometimes referred to as 'mourning' [11].

The largest discrepancy is represented by Q8 where collocated pairs possibly did not associate pair programming with Agile, while distributed pairs may have. This could be attributed to the fact that the project focused more on Scrum and less on XP.

The smallest discrepancy is represented by Q16 where there was no discrepancy. All participants scored that question with a 10. It seems that all developers experienced situations where they benefitted by putting aside their fixations and actively listened to their pair partner. Q15 is somewhat related and as expected scored high as well.

Q11 shows that while there is a preference to pair programming over pair review, the preference is not overwhelmingly strong. Both types of pairs may have felt that peer review would be more applicable than pair programming in some situations.

The social interaction questions (Q12, Q13, Q14, and Q16) scored relatively high indicating that pair programming is a highly social technique that requires attentiveness.

VII. Additional Research

This paper presented the results of performing collocated and distributed pair programming concurrently. While there are many studies on pair programming, very few compare collocated vs. distributed within the same project.

Future studies should focus on a larger subset. This paper only focused on minimal collocated pairs and minimal distributed pairs. Additionally, the use of mob programming (perhaps alongside pair programming) should also be explored. Furthermore, the equality (or inequality) of work amongst pairs could also be an interesting research area.

Even though the project team reported significant results as a result of pair programming, experiments by Nawrocki and Wojciechowski [6], Vanhanen and Lassenius [7], Arisholm et al. [8], Rostaher and Hericko [9], and Hulkko and Abrahamson [10] show little or no difference between pair programming and individual programming. Exploration as why there is such divisiveness in pair programming effectiveness may also warrant additional research.

This study did not attempt to swap whole pairs from collocated to distributed or vice versa. Further investigation into this area could highlight some transitional difficulties. This information could be invaluable at the planning stage of a project when determining which resources should be collocated or distributed.

VIII. Conclusion

This project is an indication that distributed pair programming can be effective. The results show that the following is required:

- Not surprising, both team members need to enjoy working together. Pairs that have a lot in common will naturally gravitate towards one another. To establish this working relationship it is extremely important to bring everybody together at the beginning and periodically. Throughout Project ABC, all team members (not including the client) were face-to-face once a month for a 2 day period which included sprint review, sprint planning, and sprint retrospective.
- Pairs need to receive some feedback that they are moving in the right direction. In Project ABC, emails were sent at the end of each sprint to indicate the team's velocity (Fig. 1). In most cases, the team overachieved.
- When performing the role of the 'navigator', the individual needs to ensure that the proper quality measures are enforced. Sometimes that can be as simple as reminding the 'driver' to write unit tests.
- Pairs are required to have complete trust in one another. There will be disagreements and pairs need to actively listen to options they may not have thought of. Additionally, they need to be open minded to try different techniques that may be uncomfortable.
- What may not be obvious is that the social aspect of pair programming also applies to distributed pairs. Pairs need to recognize that their counterparts may be on another time zone or participating in a cultural celebration.
- Expressing ideas is also very important. But pairs will only do so if they feel they will not be judged. Incorporating this into a team working agreement can go a long way.
- It can be difficult to let go of obsessions. When pairs are able to do this they often learn something new.

- Pair programming can be exhausting. Distributed pairs need to ensure they have enough sleep prior to starting their work day. This will allow them to actively participate throughout the day.
- There are often different levels of experience and expertise with Agile. The act of distributed pair programming can increase interest in Agile and may lead to the adoption of other XP practices.
- Tools are extremely important. Real-time or near real-time tools are necessary for distributed pair programming.

While this project proved that distributed pair programming can be effective, it also proved that collocated pair programming can also be effective. However, based on the results of the survey there is no evidence to show that distributed can be more or as effective as collocated pair programming. Keep in mind that assumes all developers are of the same level. It is also reasonable to assume that a pair or senior distributed developers can outperform a pair of junior developers.

Agile teams are likely to achieve a higher velocity by using collocated pair programming. However, if collocated is not an option, distributed pairs should be considered.

Of particular interest is the increase in velocity as pair programming teams progress. In Fig. 2, the team completed more story points in less time.

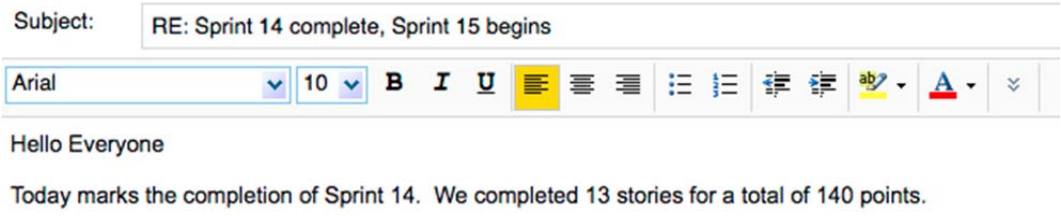

Fig. 1. Sprint 14 completion email.

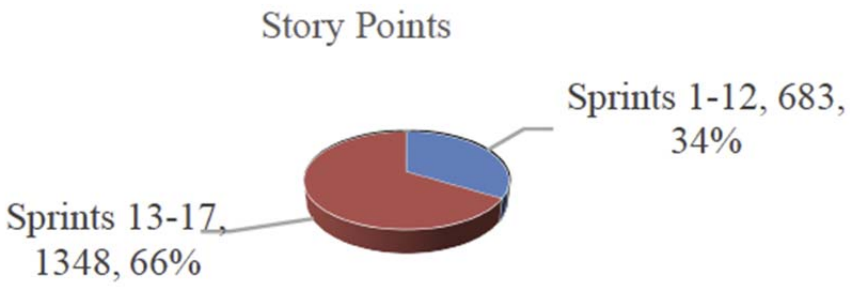

Fig. 2. Team velocity.

ACKNOWLEDGEMENT

This paper would not have been possible without the participation and hard work of the developers on Project ABC. A special thanks also goes out to the client who supported the use of Agile tools and techniques.

REFERENCES

[1] Extreme Programming. 1999 [Retrieved 11/30/2016], Available from: http://www.extremeprogramming.org/rules/pair.html.

[2] C. Ho S. Raha E. Gehringer L. Williams "Sangam—A distributed pair programming plug-in for eclipse" in Proc. OOPSLA Workshop on Eclipse Technology Exchange, pp. 73-77 2004.

[3] K. E. Boyer A. A. Dwight R. T. Fondren M. A. Vouk J. C. Lester "A development environment for distributed synchronous collaborative programming" ACM SIGCSE Bull., vol. 40 no. 3 pp. 158 2008.

[4] W. Dou, K. Hong and X. Zhang, "A Framework of Distributed Pair Programming System," 2009 International Conference on Computational Intelligence and Software Engineering, Wuhan, 2009, pp. 1-4. doi: 10.1109/CISE.2009.5363425

[5] D. Wallace, I. Raggett, J. Aufgang, Extreme Programming for Web Projects, Addison-Wesley Longman, Inc, 2003.

[6] Nawrocki, J. and Wojciechowski, A., 2001. Experimental Evaluation of pair programming. In: Proceedings of the European Software Control and Metrics Conference (ESCOM 2001). ESCOM Press, 2001, pp. 269-276.

[7] Jari Vanhanen and Casper Lassenius, Effects of Pair Programming at the Development Team Level: An Experiment, 2005 IEEE

[8] Erik Arisholm, Hans Gallis, Tore Dyba, and Dag I.K. Sjoberg, Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise, IEEE Transactions on Software Engineering, Vol. 33, No. 2, Feb 2007.

[9] Matevz Rostaher and Marjan Hericko, Tracking Test First Programming – An Experiment, XP/Agile Universe 2002, LNCS 2418, pp. 174-184, 2002

[10] Hanna Hulkko and Pekka Abrahamsson, A Multiple Case Study on the Impact of Pair Programming on Product Quality, ICSE'05, May 15-21, 2005, St. Louis, Missouri, USA.

[11] The Five Stages of Project Team Development. 2016 [Retrieved 06/15/2017], Available from: https://project-management.com/the-five-stages-of-project-team-development.

[12] 2016 State of Scrum Report. 2017 [PDF File], Available from: https://www.scrumalliance.org/why-scrum/state-of-scrum-report/2016-state-of-scrum.

[13] 11th Annual State of Agile Survey [PDF File], Available from: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2.