A Set of Practices for Distributed Pair Programming

Bernardo José da Silva Estácio and Rafael Prikladnicki
Computer Science School, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Porto Alegre, Brazil

Keywords: Distributed Pair Programming, Extreme Programming, Distributed Software Development.

Abstract: Geographically distributed teams have adopted agile practices as a work strategy. One of these practices is Distributed Pair Programming (DPP) that consists in two developers working remotely on the same design, algorithm, or code. In this paper, we describe a set of practices for DPP. In our research we seek to understand how distributed teams can use and adopt DPP in a more effective way. Based on a systematic literature review and a field study, we suggest twelve practices that can help both professionals and software organizations in the practice of DPP.

1 INTRODUCTION

The globalization experienced in recent decades has caused an impact in different domains, and consequently, in software development (Herbsleb and Moitra, 2001). In a competitive market, IT companies started to distribute their software development processes, seeking lower labor costs and better product quality, (Smite et al., 2012). Therefore, in the late 90s Distributed Software Development (DSD) was initiated. DSD is characterized when teams develop software projects and have it members dispersed across different buildings, cities, countries or continents (Herbsleb and Moitra, 2001).

Almost at the same time, in 2001, a group of 17 professionals has written the Agile Manifesto. They coined the term Agile Software Development and proposed several agile methods (Beck et al., 2001). Agile Software Development is an adaptive approach suitable for environments with volatile requirements. Agile methods have attracted interests both from academia and industry (Dyba and Dingsøyr, 2008). Extreme Programming (XP) is one of the agile methods most adopted in industry, having several practices that support software development activities (Beck, 2000; Dyba and Dingsøyr, 2008).

Pair Programming (PP) is an agile practice that is part of the XP method. In this practice, two developers cooperate to develop software using the same computer (Mcdowell et al., 2002). In PP, one developer, called the "Driver", is responsible to develop the code, controlling the mouse and the keyboard. The other developer, the "Observer", reviews the code at the same time. Previous studies showed that PP is a practice with good results such as: better code quality (less defects), and a better collaboration between the team members (Müller, 2007; Vanhanen et al., 2007).

Recent literature reported that the adoption of agile practices with distributed teams could be successfully adapted (Phalnikar et al., 2009). In this context, the concept of Distributed Pair Programming (DPP) has emerged (Baheti et al., 2002). In DPP, two developers remotely collaborate to develop software using tools that allow screen sharing and communication using audio, video, etc.

The adoption of agile practices in distributed team can introduce challenges. PP, for instance, needs a face-to-face communication and collaboration in the same physical environment (Abbattista et al., 2008). Previous research has already acknowledged that conventional agile practices need to be adjusted in globally distributed environments to address the challenges in these settings (e.g, Paasivara et al., 2009; Abbattista et al., 2008; Shrivastava et al., 2010). For this reason, we propose a set of practices for DPP to facilitate the adoption and use of this practice and better support distributed teams that use agile practices such as PP.

This paper is organized as follows: Section 2 presents some background concepts regarding DPP. Section 3 describes our research methodology. In Section 4 we present details about the SLR executed. In the Section 5 we present the details about the field

José da Silva Estácio B. and Prikladnicki R..
A Set of Practices for Distributed Pair Programming.
DOI: 10.5220/0004898503310338
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 331-338
ISBN: 978-989-758-028-4

study executed. We present the set of practices in Section 6. In Section 7 we present the final remarks, including limitations, and future work.

2 BACKGROUND AND RELATED WORK

2.1 Distributed Agile Development

The trend to adopt DSD by organizations is sometimes followed by the use of agile practices. This is motivated by projects that require faster development with volatile requirements, keeping the high quality (Shrivastava et al., 2010).

Agile methods have many benefits when combined with distributed teams. Some studies reported that practices like Continuous Integration helps to solve issues related to distributed configuration management. Sprint reviews can be used to improve external communication and share project information (Shrivastava et al., 2010). Paasivara et al (2009) say that the use of agile methods improved communication, collaboration and motivation of the team in distributed environments.

But the combination of DSD and agile methods has some challenges. Agile teams rely on intense face to face communication, both with the team and the customer (Shrivastava et al., 2010). The project documentation, different working hour and several agile practices like retrospective, stand up meeting, pair programming and others need to be adapted for the adoption of agile methods be combined with distributed teams (Shrivastava et al., 2010).

2.2 Distributed Pair Programming

Pair programming is one of the primary practices of the XP agile method. As the name suggests, is a programming practice that involves two developers working at the same computer collaboratively (Mcdowell et al., 2002). A developer behaves as driver and develops the code, controlling the keyboard and mouse. Another developer behaves as observer or navigator and is responsible for reviewing the code at the same time, prevent and identify logical and syntactical errors in the code. The both roles can be switched (Mcdowell et al., 2002).

Besides the collaboration, communication is one of the requirements of pair programming. Both developers should be in constant contact to discuss possible solutions and code errors (Mcdowell et al., 2002). Several benefits of the practice of pair programming are reported, such as: increased productivity, increased product quality due to the high number of defects found in the review of the pairs (Müller, 2007), increasing collaboration and team communication and improvement of working condition (confidence and motivation) (Vanhanen et al., 2007).

Distributed Pair Programming (DPP) is when two developers are distributed and practice PP. Some studies have investigates the effects of the DPP. Baheti (2002) reported that the benefits of PP are the same DPP, such as productivity and code quality. Another benefit of DPP is that due to its characteristics, it helps to promote the work and communication within distributed teams (Baheti et al, 2002).

In industry, Rosen et al. (2010) conducted a case study of a pilot project in two branches of a German company. They reported positive effects in communication (a more integration of the developers using DPP sessions, discussing developing issues and suggesting solutions), knowledge transfer (support the feedback and teaching of novice developers) and code quality (less defects). As negative effects were cited: distraction, lack of fulfilment of the role of DPP (both driver and observer) and conflict of goals among the developers during the session of DPP.

DPP also proved to be a practice that generates benefits in teaching programming. Positive effects were found related to learning and code quality (Zacharis, 2011). In an experiment conducted by Hanks (2005) students who used DPP had performed as well as students who were co-located with pairs, with similar grades.

3 RESEARCH METHODOLOGY

To develop a set of practices for DPP, we have followed a two-phase research methodology. The first phase was a literature review on DSD and Agile Methods. We then executed a Systematic Literature Review (SLR) about PP and DPP. The second phase involved a field study (FS) in order to evaluate the practices identified in the first phase.

3.1 Phase 1: Literature Review

In the literature review, we first did an initial ad hoc review about DSD and Agile Software Development. We analysed some of the existing

agile software developments practices and found few studies about DPP. For this reason, we planned and executed a SLR about PP and DPP. The goal was to look for more empirical evidence about the topic, including benefits, challenges, and practices. In addition, we were interested in finding empirical evidences of the usage of PP practices in the context of distributed teams.

Thus, the systematic review was guided by three research questions: (i) What is known about the use of pair programming and distributed pair programming? (ii) In which situations pair programming works? (iii) In which situations distributed pair programming works? The result of the SLR was a preliminary set of practices for DPP.

The systematic review was executed between March and November of 2012. We followed Kitchenham et al. (2007) recommendations to perform a SLR. In order to search for relevant literature, we defined the search string as "pair programming" OR "pair-programming". This definition was based on the SLR executed by Salleh et al. (2011) and a meta-analysis performed by Dyba et al. (2009). In total, the search returned 391 papers, of which 147 were considered potentially relevant based on the title and the abstract. We then read all the 147 papers, and selected 95 for a deeper reading and analysis and quality assessment. Inclusion criteria were: papers written in English, with empirical results and evidence about PP and DPP in the context of software development. The review process was conducted in pair to avoid bias. The results were organized in CiteULike.

We classified the papers into three categories: (i) Education: DPP and PP as a pedagogical tool for teaching programming, 36 papers; (ii) Practice: PP and DPP as a software development practice, 40 papers; and (iii) Tools, Models, Frameworks: studies that describe models, frameworks and tools to support PP and DPP), 19 papers. Most of the studies we found are related to PP (only 22 papers were related to DPP). In addition, there are few papers describing case studies about the adoption of DPP in industry. Most of studies about DPP were concentrated in tool proposals.

3.2 Phase 2: Field Study

Based on the preliminary set of practices, we planned a field study with software development professionals in industry in order evaluate the practices found in the literature. We used a guide for semi-structured interviews with open-ended questions. The interviews were conducted between November of 2012 and January of 2013.

We executed a field study following the recommendations proposed by Oates (2006). We did content and face analysis with a senior researcher in agile methodologies. Based on his feedback, we improved the data collection instrument. After this step, we executed a pre-test in order to know the average time for each interview, and how clear the questions were. For data analysis we use content analysis.

The field study involved ten software projects from eight different companies. All projects used DPP. Table 1 summarizes the projects analysed. We conducted interviews with 14 professionals who were selected based on their experience with software development projects and DPP. The questionnaire (Table 2) was developed in themes such as variables (code quality, productivity, etc), DPP aspects (infrastructure, tools, etc.), benefits, challenges and opinion (suggestions).

Table 1: Summary of project analysed.

Org.	Proj.	N.	Countries involved	Language
A	1	4	Brazil, India, USA	English
A	2	2	Brazil, India, USA	English
B	3	1	Brazil, India, Russia and China	English
B	4	1	Brazil, India, China	English
C	5	1	Brazil	Portuguese
D	6	1	USA	English
E	7	1	Macedonia, South Africa	English
F	8	1	Brazil, USA	English
G	9	1	Poland, UK	Polish
H	10	1	Brazil	Portuguese

The subjects had an average of 8.2 years experience in software development. As for experience with DSD, the average is 3.7 years. All respondents had experience with PP, with an average of 3.8 years, and an average of 2 years of experience with DPP (6 months was the minimum DPP experience reported). The project teams used different languages to communicate: 7 projects have used English, two Portuguese, and one project used Polish.

Table 2: Questionnaire applied in the field study.

Theme	Question
Variables	What kind of effect DPP brought to code quality? Why?
	What kind of effect DPP brought to the team productivity and communication? Why?
	What kind of effect did DPP brought to the difference of knowledge between the pairs? Why?
	Is there any other variable affected by the use of DPP? Which variables and what are the effects observed?
DPP aspects	Is there any company guideline for using DPP?
	What infrastructure and methods have been used with DPP?
	What tools are used for DDP? Is there any specific development tool for DPP?
	Is there a facilitator or leader (coach) to support this practice in the company?
	Is there any criterion established for arranging the pairs in DPP?
	Who is responsible for choosing the pairs?
	The difference of knowledge and experience between the pairs is/was a problem? Why?
Benefits	Regarding the knowledge transfer between the pairs, were there benefits from the use of DPP? Which?
	Regarding the task execution time, were there benefits from the use of DPP? Which?
	In your opinion, regarding the motivation of the developers, were there benefits from the use of DPP? Which?
	Have you seen other benefits from the use of DPP? Which?
Challenges	In your opinion, what were the communication challenges found in DPP and how were they solved??
	In your opinion, what were the collaboration challenges found in DPP and how were they solved?
	What other challenges were identified in DPP? And how were they solved?
Opinion	Based on your experience, how do you compare the distributed development performed with and without the use of DPP?
	From your experience with projects using DPP, which would be your suggestions to complement the DPP environment and the practice?

4 A SET OF PRACTICES FOR DPP

Practice is the actual application or use of an idea, belief, or method, as opposed to theories relating to it (Oxford, 2010). We organized the set of practices, following four steps: (i) Based on the SLR results (literature), we propose a set of preliminary practices for DPP, also trying to understand if the recommended PP practices are applicable for DPP; (ii) Assess if the practices identified in the SLR (literature) are corroborated by the practices identified in the field study (industry); (iii) List the practices obtained only from the SLR (literature); (iv) List the practices obtained only from the field study (industry). The twelve practices are presented in Table 3.

Table 3: Practice and source.

Practice	Source
1. Use a guideline for DPP adoption	SLR
2. Conduct a meeting alignment before DPP sessions	SLR
3. Train the team on PP and DPP	SLR
4. Adopt a specific infrastructure	SLR, FS
5. Young professionals should form pairs both for easy and complex tasks	SLR (PP), FS
6. Identify a DPP leader within the team	SLR, FS
7. Define and use a specific tool for DPP	SLR, FS
8. Plan frequent meetings	SLR, FS
9. Provide feedback during DPP sessions	FS
10. Plan short DPP sessions with frequent breaks	FS
11.Plan a pilot project before adopting DPP	FS
12. The driver must narrate actions during the sessions	FS

Five of the practices were found both in the field study and SLR. One practice is a PP practice that could be adapted for DPP. Four practices were identified in the field study and three practices were identified in the literature SLR.

4.1 P1: Use a Guideline for DPP Adoption

We identified this practice in our systematic literature review. According to Canfora et al. (2006), the use of a guideline (behavioural protocol) for DPP can facilitate the understanding of the practice. This guideline should have details about frequency of switch between the pairs, activities of each paper (Driver and Observer), pair training criteria, type of

tasks which recommends the use of PP, rules for using the infrastructure and other aspects.

The use of this guideline is recommended in the adoption phase of DPP, mainly when the team has no experience with PP before. The guide could help the developers to know how to perform each role. For young developers it is important to clearly distinguish between the Observer and the Driver, because in distributed settings people tend to work asynchronously on different tasks instead of as pairs in the same task. For this reason they need to know how to behave in this context (Canfora et al., 2006).

4.2 P2: Conduct a Meeting Alignment Before DPP Sessions

In the literature, Rosen et al. (2010) reported that sometimes the pairs have conflicts in terms of goals to achieve, and this is a challenge for DPP. To minimize this challenge, frequent short meetings can be conducted, so the pairs can align the goals that will be discussed in the session. The meeting alignment avoids discussions during the session that do not add to the goal of the task executed in pairs.

4.3 P3: Train the Team on PP and DPP

The literature also reported that besides a guideline, it is important to have an adequate training to better understand the practices and responsibilities of each role (Canfora et al., 2006). As an example, Rosen et al (2010) reported as a challenge the failure to correctly execute the roles of DPP (driver and observer). The authors also reported as a practice the proper training of the team on PP and DPP (especially those who have never had experience with PP), in order to had more technical knowledge about the practice.

4.4 P4: Adopt a Specific Infrastructure

This practice refers to the need to have a infrastructure to support DPP. In the literature, Bevan et al. (2002) reported the importance of having a specific room for the team. This would allow to identify the proper infrastructure and equipments for the teams, and not to disrupt other team members who are not practicing DPP.

In the field study, some respondents reported that the use of large TVs and monitors also help to create a sense of physical proximity between the pairs. Other infrastructure aspect related was the Internet link. One of the respondents from project 3 said:

"The Internet connection in India was not good and it had a delay when we were at a DPP session. At certain times, the developer narrated what he was doing, but the mouse cursor or keyboard had not performed that action. "

This is what the respondent from project 7 said about the use of a webcam:

"In DPP we cannot know what the observer is doing, sometimes the developer seemed disperse. Thus, I believe it is very important to use the webcam to try to keep the focus of the DPP."

4.5 P5: Young Professionals Should Form Pairs Both for Easy and Complex Tasks

This practice is related to the formation of the pairs. Dyba et al. (2009) found that the difference of experience between the developers generates different types of results. The authors reported that the use of two senior professionals is recommended only in cases where the task has a very high critical level. The less experienced professionals (young, or junior) must always pair in order to ensure knowledge sharing. In previous studies, code quality has improved when the coding activity was executed in pairs formed by young and more experience professionals.

Some projects analysed in the field study indicated that the criterion to form the pairs was the experience of the developers. When a young developer started in the team, he already attended pair programming sessions. This is illustrated by a quote from a respondent from project 3:

"One of the main challenges that we noticed was the difference of knowledge regarding the business. Then, the pairs were formed by a developer with a lot of knowledge in the business with another developer new to the team, in the learning process."

Other respondent from project 6 said:

"The criterion to form the pairs is the level of experience, because this enables greater knowledge sharing between the team members. In addition, the task does not belong to only one pair or group."

4.6 P6: Identify a DPP Leader within the Team

This practice refers to the presence of a DPP leader (coach) in the team. In the literature, Hannay et al. (2010) reported that the practice helps avoiding impediments between the pairs and supports the developers to ask their questions about the practice. The presence of a DPP leader who acted as a Coach

was identified in the field study as a practice that helps in resolving technical questions and conflicts. Furthermore, the DPP leader was responsible for the formation of the pairs. A respondent from project 6 said:

"I perform this role; I answer all questions of the team members regarding the DPP practice and promote it in the organization. Sometimes, we promote training on DPP adoption and constantly encourage feedback from employees. The leader role has stimulated the adoption of the practice among the team members and has helps with the challenges that we have observed. "

4.7 P7: Define and Use a Specific Tool for DPP

In the literature, Canfora et al. (2006) says that the main problem in DPP is the lack of an appropriate tool to support the practice. He suggested that a tool for DPP must be integrated with a configuration system management strategy. Rosen et al. (2010) reported that a specific tool for DPP facilitates the use of a single developer environment, without the need to toggle between windows, increasing the agility. The author also states that a specific tool that shares the video screen helps in reducing distraction among developers. In our field study, people interviewed reported the use of several tools, as illustrated by the quote of the respondent from project 1:

"We try to use several tools for DPP, but none of them was stable enough to be used in the project. So we opted for common chat software, but I believe that it is not ideal; I believe that a tool that is integrated with our development tools can help more in terms of productivity of the team."

In six of the projects (4, 5, 6, 7, 9, 10) the respondents reported the use of Skype for DPP. Two of them (6, 7) used Skype with Tmux. Two projects used Skype and VNC (1, 8) and one project used Microsoft Communicator (3). One project (2) reported to use a specific DPP tool, called Saros, which is a plugin of Eclipse IDE (Salinger et al, 2010).

4.8 P8: Plan Frequent Meetings

The literature reported that the distribution between the pairs tends to decrease the level of conversation within the team; as a consequence, there is a lack of a common knowledge and understanding of the project (Canfora et al., 2006). Canfora et al. (2006) also says that team meetings can be scheduled during all phases of the project, especially at the beginning. In the field study, Respondents also confirmed the adoption of this practice, by conducting technical meetings and design sessions before the DPP sessions, in addition to the daily meetings proposed in the Scrum method (Schwaber, 1995).

4.9 P9: Provide Feedback during DPP Sessions

In the field study, the need for feedback during each session was reported as an important source of identifying challenges and improvements for the practice of DPP. During the interviews, the respondents said that it is important to ask basic questions to stimulate feedback such as: *"What is your opinion about the DPP session?"; "What can be improved in the DPP environment?";"Which strategies can be adopted to improve the DPP session?"*

Feedback is an important principle of the XP method and other agile methods and practices. For this reason, it is important to stimulate feedback during the DPP sessions, and not only at the end. If the team have a DPP leader, he can encourage others team members to share their experiences and feedback.

4.10 P10: Use Short DPP Sessions with Frequent Breaks

Most of the respondents in the field study reported that DPP is a practice that requires more effort from the team. To minimize this, a practice that was recommended is the use of short sessions with frequent breaks. The result of using this practice is to reduce the effort, and help to keep the focus in the task. One strategy mentioned was the use of time management techniques, such as the Pomodoro Technique (Nöteberg, 2009). The respondent from project 3 said:

"DPP takes more effort than PP and collocated development; it requires more focus and less distraction. Collaboration is more intense. I believe that with shorter sessions, this challenge can be reduced."

4.11 P11: Plan a Pilot Project before Adopting DPP

In the field study the respondents indicated that before the adoption of DPP is important to understand the details of the environment. In the

field study, some of the respondents reported that it is important to plan and execute a pilot project using DPP before the practice is implemented. The pilot project has the goal of identifying challenges and planning strategies to improve the DPP environment, as well as helping employees to become familiar with the practice.

4.12 P12: The Driver Must Narrate Actions During the DPP Sessions

Another practice that was identified in the field study is regarding the role of the driver. The respondents said that during the DPP session the driver must constantly narrate his actions in the code. The adoption of this practice involves the reduction of periods of silence that are harmful to DPP. Another benefit is the monitoring of the observer, which is also stimulated to seek communication during the session. The respondent from project 2 said:

"One of the challenges we had with DPP is when another developer was in silence during the session. This type of behavior affects the practice, so when we identified it, we encourage the conversation. DPP is a practice that works well with a strong communication.

5 FINAL REMARKS

In a recent study, Paasivara et al. (2009) reported that the use of agile practices is part of a strategy that companies use in order to minimize the challenges face by distributed teams. DPP is an agile practice that consists in a variation of PP with distributed teams (Baheti et al., 2002). Previous study has found that DPP have benefits for distributed teams such as communication and knowledge transfer, but also has challenges such as distractions, the need for specific infrastructure, and conflict of goals between the pairs during a DPP session (Rosen et al., 2010).

The main motivation for the development of this research was at the same time the lack of studies about the use of DPP in industry and the opportunity to better understand how DPP could help companies facing DSD challenges. Based on the evidence of existing literature and a field study, the set of practices proposed in this paper has the purpose of helping software development organizations to adopt or improve the use of DPP in order to increase the chance to succeed with this practice. In addition, it is an attempt to contribute to the adoption of agile practices in the context of DSD.

5.1 Limitations and Future Work

As any other empirical study, this study also has limitations. Regarding the systematic literature review, the studies selected were all collected from online libraries. We chose these libraries based on past experiences of other researchers in conducting SLR. The bias of the researchers during the analysis of the papers was minimized by a peer-review recommended by Kitchenham et al. (2007). Regarding the field study, one limitation is related to the number of companies studied, limiting the generalization of the results.

As future work, we plan to evaluate the set of practices in industry and continue to evaluate and refine the proposed practices, and also identifying new practices to be adopted by distributed agile teams.

ACKNOWLEDGMENTS

We are thankful to all the respondents who have kindly accepted our invitation and contributed to this research. We would like to thank one of the companies for having sponsored this research under the Brazilian Law 8.248/91. This research is also part of the agreement between ThoughtWorks and PUCRS. We also thank CNPq (309000/2012-2).

REFERENCES

Abbattista, F.; Calefato, F.; Gendarmi, D.; Lanubile, F. 2008. Incorporating social software into distributed agile development environments. In: *Automated Software Engineering - Workshops, ASE Workshops*, pp. 46,51, 15-16. IEEE: L'Aquila, Italy.

Baheti, P.; Gehringer, E.; Stotts, D. 2002. Exploring the efficacy of distributed pair programming. In: *Extreme Programming and Agile Methods—XP Agile Universe*, pp.387-410. Springer: Chicago, USA.

Beck, K. 2000. Extreme Programming Explained: Embrace Change. New York. Addison Wesley, 2nd edition.

Beck, K.; et al. 2001. Manifesto for Agile Software Development. Accessed in: www.agilemanifesto.org.

Bevan, J.; Werner, L.; McDowell, C. 2002. Guidelines for the use of pair programming in a freshman programming class. In: *Conference on Software Engineering Education and Training*, pp.100-107, IEEE: Ottawa, Canada.

Canfora,G.; Cimitle, A.; Visaggio, C.;DiLucca, G. 2006. How distribution affects the success of pair programming. In: *International Journal of Software Engineering and Knowledge Engineering*, pp. 293 - 313. World scientific.

Dyba, T.; Arisholm, E.; Sjoberg, D.I.K.; Hannay, J.E.; Shull, F. 2009. The effectiveness of pair programming: A meta-analysis. *Information Software and Technology*, vol. 51-7, pp. 1110-1122. Elsevier.

Dyba, T.; Dingsøyr, T. 2008. Empirical studies of agile software development: A systematic review. *Information Software Technology*, vol. 50-10, Ago-2008, pp. 833-859. Elsevier.

Hanks, B.,"Student performance in CS1 with distributed pair programming". 2005. In: *Proc. of the Special Interest Group on Computer Science Education,* pp. 316-320. ACM: Saint Louis, USA.

Hannay, J.; Arisholm, E.; Engvik, H.; Sjoberg, D, 2010. Effects of personality on pair programming. *Transactions on Software Engineering*, vol. 36-1, Fev-, pp.61–80. IEEE.

Herbsleb, J.; Moitra, D. 2001. Global Software Development. *IEEE Software*, vol. 16-2, Mar- Abr 2001, pp. 16-20. IEEE.

Kitchenham, B; Charters, S. 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report, p. 57. Keele University and Durham University.

Mcdowell, C.; Werner, L.;Bulock,H. ; Fernald, J, 2002. The effects of pair-programming on performance in an introductory programming course. In: *Technical symposium on Computer science education*, pp. 38–42. ACM: Cincinnati, USA.

Müller, M, 2007. Do programmer pairs make different mistakes than solo programmers?. *Evaluation and Assessment in Software Engineering*, vol.80-9, Set-2007, p.1460–1471. Elsevier.

Nöteberg, S. 2009. Pomodoro Technique Illustrated: The Easy Way to Do More in Less Time. Raleigh. Pragmatic Programmers.

Oates, B. J, 2006. Researching information systems and computing. London. Sage.

Oxford American Dicitionary, 2010. Oxford. Oxford University Press, 3rd edition.

Paasivaara, M.; Durasiewicz, S.; Lassenius, C, 2009. Using Scrum in Distributed Agile Development: A Multiple Case Study. In: *International Conference on Global Software Engineering*, pp. 195-204. IEEE: Limerick, Ireland.

Phalnikar, R.; Deshpande, V. S.; Joshi, S. D., 2009. Applying Agile Principles for Distributed Software Development. In: *International Conference on Advanced Computer Control*, pp.535-539, IEEE: Singapore.

Rosen, E.; Salinger, S.; Oezbek, C, 2010. Project Kick-off with Distributed Pair Programming. In: *Workshop of Psychology of Programming Interest Group*. Madrid, Spain, 15p.

Salleh, N.; Mendes, E.; Grundy, J., 2011. Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. In: *Transactions on Software Engineering*, vol.37-4, Jul-Ago. 2011, p. 509–525. IEEE.

Schwaber, K, 1995. The Scrum Development Process. In: *Conference on object oriented programming systems, languages and applications* (OOPSLA), 1995, pp. 117-134. Springer: Austin, USA.

Shrivastava, S; Date, H, 2010. Distributed agile software development: a review. *Journal of Computer Science and Engineering*, vol 1-1, May 2010. KIISE.

Smite, D. ;Wohlin, C .; Galvina, Z. ; Prikladnicki, R, 2012. An Empirically Based Terminology and Taxonomy for Global Software. *Engineering. Empirical Software Engineering: An International Journal*. Springer.

Salinger, S.; Oezbek, C.; Beecher, K.; Schenk, J., 2010. Saros: an eclipse plug-in for distributed party programming. In: *Proceedings of the ICSE Workshop on Cooperative and Human Aspects of Software Engineering* (CHASE '10). ACM: Cape Town, South Africa.

Vanhanen, J.; Lassenius, C.; Mantyla, M.V, 2007. Issues and Tactics when Adopting Pair Programming: A Longitudinal Case Study. In: *Software Engineering Advances*, pp. 25-31. IEEE: Cap Esterel, France.

Zacharis, N, 2011. Measuring the effects of virtual pair programming in an introductory programming java course. *IEEE Transactions on Education*, vol. 54-, Fev-2011, pp. 168–170.