Evaluating the Use of Pair Programming and Coding Dojo in Teaching Mockups Development: An Empirical Study

Bernardo Estácio
Pontíficia Universidade Católica
do Rio Grande do Sul
bernardo.estacio@acad.pucrs.br

Natasha Valentim
Universidade Federal do
Amazonas
natashavalentim@icomp.ufam.edu.br

Luis Rivero
Universidade Federal do
Amazonas
luisrivero@icomp.ufam.edu.br

Tayana Conte
Universidade Federal
do Amazonas
tayana@icomp.ufam.edu.br

Rafael Prikladnicki
Pontíficia Universidade Católica
do Rio Grande do Sul
rafaelp@pucrs.br

Abstract

Collaborative programming is an important pedagogical tool in computer science higher education. In this context, Pair Programming has been established as an effective practice for teaching programming. In addition, Coding Dojo has recently emerged as a collaborative group practice that uses Pair Programming as a mechanism to allow everyone to participate. However, both Pair Programming and Coding Dojo are rarely used in different types of programming tasks such as front-end programming tasks. In this paper, we present an empirical study comparing Pair Programming and Coding Dojo in the teaching of mockups development. Our goal was to evaluate both practices regarding three dimensions: motivation, user experience and learning perceived by students. The results showed that Pair Programming was well accepted by the students with positive results in all three dimensions. Moreover, although Coding Dojo has presented positive results in the leaning process, students reported several challenges related to motivation and user experience.

1. Introduction

Collaborative programming is an important pedagogical tool in computer science higher education. Students not only need to know how to develop a software that can be easily comprehended by others, but they also need to learn how to develop software with others, learning to work in a team or how to be part of a team [2]. For this reason, collaboration is one of the key aspects in the teaching of software development, providing a process of innovation and ideas generation among the programmers [1].

In this context, Pair programming (PP) has been established as an effective practice for teaching programming. PP is one of the main practices from Extreme Programming (XP), one of the most well known agile methods [3]. PP promotes collaboration between two developers. By collaborating in pairs, such practice provides programmers with an enjoyable environment [4] that promotes academic performance [5]. The results can, in some cases, be reflected by higher grades [6], and confidence increase [4], when compared to individual programming.

Over the years PP has been established as a strong foundation [7] and an effective pedagogical tool in higher education [8]. Moreover, Cockburn and Williams [9] report that PP is an effective pedagogical tool due to its capability of increasing learning capacity.

More recently, Coding Dojo has also emerged as a collaborative programming practice, providing a non-competitive environment of group participation and learning. There are several variants of Coding Dojo [10] and one of them is called Randori. This variant adopts PP to promote the engagement of all participants.

Few empirical studies explore the evidences of Coding Dojo in higher education, but the initial results are promising. Heinonen et al. [11] and Da Luz et al. [12] report positive results on the leaning of agile practices such as Test Driven Development (TDD) and Pair programming. Coding Dojo also proved to be an enjoyable practice for the students and a space to share knowledge [10].

Most of the studies involving Pair Programming and Coding Dojo are in the introductory programming courses, dealing with programming tasks. Empirical research rarely explores pair programming and coding dojo in courses in which students are exposed to tasks such as design tasks or front-end programming tasks [8].

As an example, Canfora et al. [13] report an experiment with the use of Pair Programming in design tasks. The results showed benefits in time and the quality of the work. The two controlled experiments

1530-1605/15 $31.00 © 2015 IEEE
DOI 10.1109/HICSS.2015.602

performed by Lui et al. [14] reported that the pairs outperformed the solo programmers. Nonetheless, we are not aware of any study aiming to investigate the performance of using Coding Dojo in developing mockups.

In order to explore the wider applications of Pair Programming and Coding Dojo in the development of front-end programming tasks, we planned and conducted an empirical study in a Software Development Analysis and Design course. Our goal with this study was to evaluate the learning, motivation and user experience of Pair Programming and Coding Dojo in the development of mockups. In software development, mockups represent a prototype of a software feature that enables the testing of its design. Mockups are important since they can be used to gain feedback of the users regarding the system's usability [15].

Our study offers the following main contributions:

- An empirical comparative analysis between Pair Programming and Coding Dojo in the context of higher education regarding learning, motivation and user experience of the use of mockups for software development;
- Empirical evidence about the practice of Pair Programming and Coding Dojo in the development of front-end programming tasks, such as the development of mockups.

The paper is organized as follow: in Section 2 we present the background for this research. In Section 3, we describe our research methodology, presenting the settings of the conducted empirical study. Then, in Section 4, we present the results of the study, while Section 5 shows its threats to validity. In Section 6, we discuss the study results. Finally, in Section 7 we draw the conclusions and the next steps of this research.

2. Background

2.1. Pair Programming

As the name suggests, Pair Programming (PP) is a practice that involves two developers working at the same computer collaboratively [4]. In a PP session, a developer acts as driver and develops the code, controlling the keyboard and mouse. Another developer acts as the navigator and is responsible for reviewing the code, preventing and identifying logical and syntactical errors in the code. During a PP session the pairs can switch the roles [4]. PP is often related with agile practices, because it has gained popularity as a primary practice from Extreme Programming [3].

Several previous controlled experiments aimed to explore the efficiency of pair programming. In that context, PP presented many benefits over solo programming such as the quality of the developed software (less defects) [4], knowledge transfer, productivity [17, 18] and also enjoyment (motivation and satisfaction) among the developers [5].

Specifically, in the context of higher education, pair programming consolidates the benefits in the learning process, promoting the confidence and social interaction between the students [8]. Nagappan et al. [16] reported that students who adopted pair programming were more self-sufficient, generally perform well on projects and exams and were more likely to complete the course than the students who practiced solo programming.

2.2. Coding Dojo

Coding Dojo is a session where a group of participants gather to practice programming together [11]. The main goal of Coding Dojo is to promote a safe learning environment: collaborative and with no competition [10]. In the literature, Coding Dojo is also related to the learning of agile practices such as Test-Driven Development (TDD), refactoring and pair programming [10].

There are several types of Coding Dojo, and one of the variants is called Randori, in which Pair Programming is the main mechanism that enables participation in the group [11]. In a Randori session, one participant acts as a driver and the other one as a navigator. The remaining acts as an audience that pays attention to the pairs. The audience is able to participate only with the agreement of the pairs in a coordinated way. Each round, the driver moves to the audience, the navigator turns into the driver and someone of the audience start to act as a navigator. Every participant acts at least one time as a driver and as a navigator.

Few studies explore empirical evidence about Coding Dojo. Sato et al. [10] reported that Coding Dojo impacts in the learning process and also present a set of lessons learned related to the environment and different variants of Coding Dojo in a computer science education setting. Da Luz et al. [12] reported a Randori experience that aimed at investigating the learning of TDD through Coding Dojo. Their results showed that the session helped learning TDD and that pair programming supported the leveling of the group.

Heinonen et al. [11] conducted Coding Dojo sessions into the agile part of an undergraduate software engineering course. The survey filled by the participants presented good results in the learning of TDD. Most of the students saw the sessions as a relaxing and non-competitive environment. A

drawback reported by the students was that the 5 minutes time box of the sessions was too short.

3. Research Design

In this section, we described the design of our empirical study. Our planning was inspired by the suggestions proposed by Wohlin et al. [19] for conducting empirical studies. The authors recommend, for example, the randomization of the sampling and the balancing of each group of subjects.

3.1. Goal

The main goal of this study is to investigate the use of Pair Programming and Coding Dojo in the teaching of mockups development in a computer science course. The detailed goal is structured as follows:

Analyze: Coding dojo and Pair programming in teaching of mockups development;
For the purpose of: Characterize;
With respect to: motivation, user experience and learning and speed comparing with pair programming;
From the point of view: of the researchers and the students;
In the context of the development of mockups by undergraduate students;

3.2. Subjects

The empirical study was conducted in the first semester of 2014 in an Analysis and Design class within a computer science course. This is a 3rd year class of the course and has as prerequisite classes of Introduction to Software Engineering and Introduction to Programming Language. Seventeen (17) students participated in the empirical study.

In order to participate in the study, all the students signed a consent form and filled out a characterization form with objective questions to inform us about their expertise in the topics related to the study: (a) their experience in programming; (b) their expertise in Qt (Qt is a multiplatform development framework which is gaining popularity, and was applied during the development of the mockups); (c) their expertise with Pair Programming; and (d) their expertise with Coding Dojo.

We collected the data characterization form from each student and ranked them into having: none (N), low (L), medium (M) and high (H) experience for the respectively expertise identified. For instance, regarding programming and Qt expertise, the subject was characterized as having: (a) No experience, if he/she have never had contact with the framework nor practiced it; (b) Low experience, if he/she had had contact with programming only in the classes or reading a support material; (c) Medium background if he/she had contact with programming in an academic project; or (d) High if he/she had experience in the industry. Similarly, the expertise for Pair Programming and Coding Dojo was assigned according to the number of sessions in which the subject had worked in such activities: (a) No experience; (b) Low: 1 session; (c) Medium: more than 1 to less than 4 sessions; and (d) High: more than 4 sessions.

After ranking the participants' experience, we divided them into two balanced groups (Pair Programming and Coding Dojo). By balancing, we mean that we avoided that one team had more experienced students than the other in order to avoid biased results of a team performing better in the assigned tasks. Table 1 shows each of the groups defined and their expertise and the expertise of each of its members.

Table 1. Expertise per participant in each group

Pair Programming Group								
	Pair 1		Pair 2		Pair 3		Pair 4	
Subject ID	P1	P3	P2	P4	P5	P8	P6	P7
Programming	H	N	L	N	L	N	M	N
Qt	L	N	N	N	L	N	N	N
Pair Programming	N	N	M	N	M	M	N	N
Coding Dojo	N	N	N	N	N	N	N	N

Coding Dojo Group									
Subject ID	CD1	CD2	CD3	CD4	CD5	CD6	CD7	CD8	CD9
Programming	N	L	N	L	N	L	N	L	M
Qt	N	N	N	N	N	N	N	N	L
Pair Programming	H	M	N	M	N	L	N	N	N
Coding Dojo	N	N	N	N	N	N	N	N	N

Beside the students, two researchers acted as instructors and observers, supporting each group in their respective rooms. Also, two other researchers acted as monitors helping in preparing the materials and locations in which the study would take place.

3.3. Procedures and materials

All subjects had training in the Qt framework, specifically on how to develop mockups and the transitions among them. This was undertaken since they would be able to develop the necessary code for implementing functional mockups, which could be used for showing the interaction of the software's graphical user interface.

In the day of the study, the students were equally distributed into the teams based on the results of the characterization form delivered previously. Each group went to a different room in order to avoid the bias of communication among the students from each group. Each instructor gave a 10-minute talk in each room about the respective practice (either Coding Dojo or Pair Programming). From the 17 subjects, 9 students were assigned to the Coding Dojo group and 8 to the Pair Programming Group. We took this decision based on the number of subjects needed to form the pairs for the Pair Programming group.

The objects of study were the mockups from a real mobile Web application called Dona Know (http://donaknow.aondefui.com/). Dona Know provides a list of events (e.g. concerts, shows, social events, others) for public consultation. Since Dona Know is currently under development, a set of real mockups was made available for this research. The development team at Dona Know wanted to have the graphical user interface developed, so they would be able to perform tests with end users and verify if their user interface proposal met the users' needs.

All subjects received pictures of five of the mockups from the Dona Know application and a model of the interaction flow that they should develop. This subset of five mockups was chosen by the Dona Know development team since, according to them, it was critical for showing the application's functionalities. Each group had a time box of 10 minutes to switch the pairs. The pairs in the Pair Programming group had been defined based on the balancing of the group (see Table 1). Thus, an experienced student along with a less experienced student formed a pair, in order to increase learning. In the Coding Dojo group, the sequence of the pairs was made by convenience (at the students' choice). However, the audience was also able to participate in a coordinated way, if the pairs that had the control at the time of the session agreed to the intervention.

The study lasted approximately two hours for each group simultaneously in different rooms. The Pair Programming group carried out 10 sessions and the Coding Dojo group carried out 11 sessions.

At the end of study, each student answered a questionnaire and sent the code implementing the mockups to the instructors. All students collaborated in this process with the study and no data were discarded.

The post-study questionnaire and the evaluation were adapted from Wangenheim et al. [20]. The study of Wangenheim et al. [20] has been executed in the context of agile serious games. We have selected this study, because it has a specific framework to assess the sub-components of: the learning process, user experience and motivation.

The post-study questionnaire consist of 12 fixed items divided into 3 three sub-component (Motivation, User Experience and Learning) and 8 dimensions on a Likert scale with response alternatives ranging from strongly disagree (−2) to strongly agree (2).

As our study has the focus in agile practices and not in agile games, we have kept or adapted the arguments and removed some dimensions or arguments from dimensions that we were not the focus of our study. The dimensions and respective arguments that we have adapted and kept were listed in Table 2.

Table 2. Arguments adapted/kept from Wangenheim et al. [20]

Motivation		
Attention	There was something interesting of the practice that got my attention.	Adapted
Relevance	The way of the practice works suits my way of learning.	Adapted
Confidence	As I worked on the practice, I felt confident that I was learning.	Adapted
	It was easy to understand practice and start using it.	Adapted
User Experience		
Competence	I had positive feelings of efficiency during his practice.	Adapted
Fun	I had fun with the practice.	Adapted
	I would recommend this practice to my colleagues.	Adapted
	I would like to play this practice again.	Adapted
Challenge	This practice is properly challenging for me, the tasks are not too easy nor too difficult.	Adapted
Social Interaction	I had fun with the group.	Kept
	The practice promotes moments of cooperation between the players.	Adapted
Learning		
Long-term learning	The experience with the practice will improve my performance in future working life	Adapted

In relation to the learning, questions have been added to elicit the perceived knowledge level before

and after the practice in respect of the concepts taught: Mockups, Interaction of Mockups and Qt, based on Bloom's taxonomy [21]. In this question the students give a grade ranging 1 to 5 to each question based on t perception of their evolution of learning in the concepts taught before and after the practice.

We also customized the post-study questionnaire, adding two open questions in order that students could explain benefits and disadvantages about Pair Programming and Coding Dojo.

We have collected all the data from the post-study questionnaire and execute a thematic analyze in relation to each category with the information collected. We have used spreadsheets to organize the quantitative information, generating automatized graphics.

4. Results

We have gathered data from the likert questions in the post-study questionnaire and qualitative data from the open questions and additional comments. Figures 1 and 2 present the results of the Pair Programming and Coding Dojo groups regarding motivation and user experience. Figure 3 shows the results of the long-term learning Figures 4 e 5 present the grades of the learning dimension based on Bloom's [21].

4.1. Motivation

Overall, students perceived a positive contribution of the Pair Programming (see top of Figure 1) and Coding Dojo (see bottom of Figure 1) practices for engaging them into learning about the development of mockups. Regarding the attention dimension, both practices presented positive results, motivating students. The main aspect that got the attention of the students in both practices was the interaction between the students.

Regarding the relevance dimension (pertinent to be adopted, used), Pair Programming presented positive results with most of the students in relation to acceptance of the practice, but Coding Dojo was not widely accepted, five students reported that the practice did not suit with their way of learning. In Coding Dojo, the subject 1 said: "Coding Dojo is not suitable to my way of learning due to many different ideas".

Concerning the confidence dimension, the items related to understanding and ease of use of the practice, both Pair Programming and Coding Dojo presented positive results. Pair Programming presented most positive results about the student's impression of confidence in learning. On the other hand, in the Coding Dojo group most of the students reported a decrease in the confidence of learning.

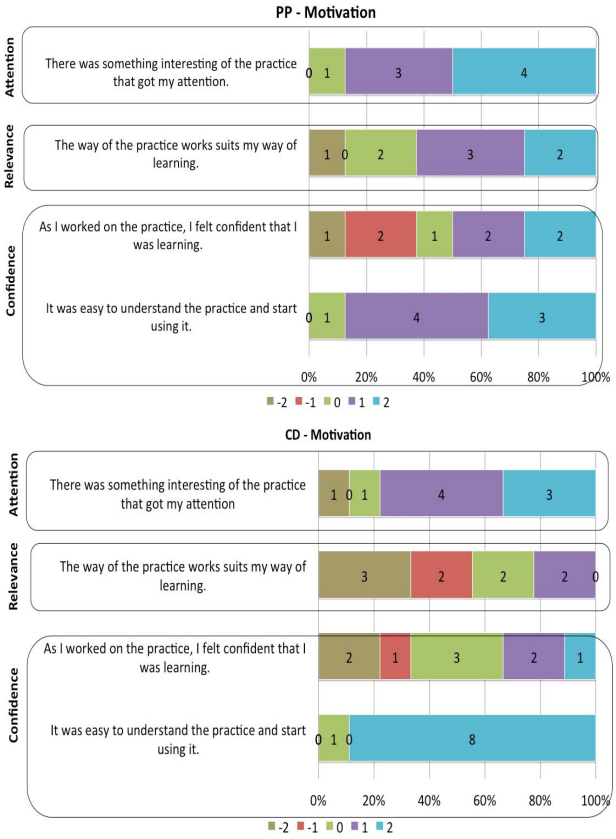

Figure 1. Frequency diagrams about the Motivation dimension in Pair Programming and Coding Dojo group

4.2. User Experience

The user experience (the perception or reaction of the student in relation to the adoption of the practice) from the students presented positive results in Pair Programming (see top of Figure 2). Coding Dojo (see bottom of Figure 2) had diverse results in some items of each dimension.

Regarding the competence (the ability to be efficiency with the practice), most of the students expressed positively their belief that the Pair Programming has been an efficient way to learn.

In the Coding Dojo group, the feedback was more diverse. Regarding "fun", in the Pair Programming group all the students reported that they want to use the practice again and most of them said that they had fun and would recommend the practice to a colleague. On the other hand, the results of the Coding Dojo group were not positive at all, specifically in the recommendation and use the practice again.

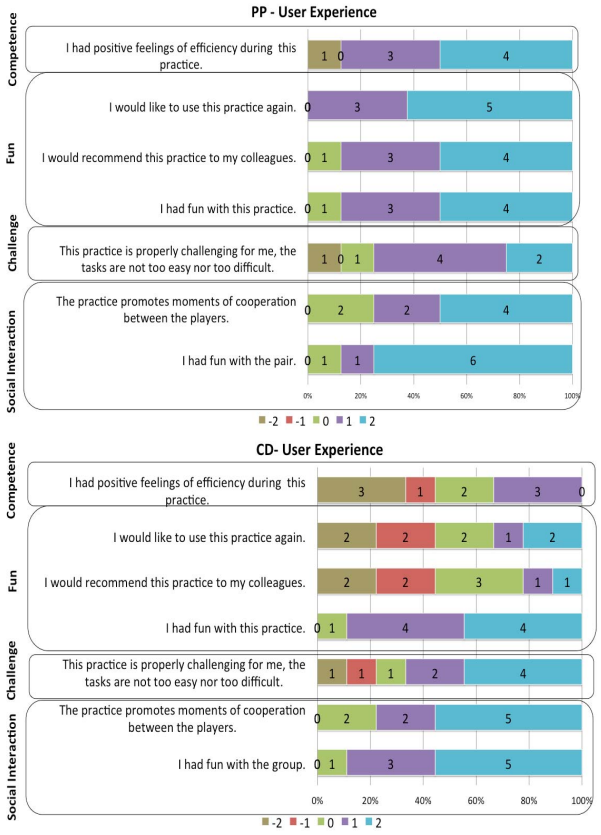

Figure 2. Frequency diagrams about the User Experience dimension in Pair Programming and Coding Dojo group

Concerning the challenge of the use of Pair Programming and Coding Dojo, both practices showed not to be difficult for each group. The social interaction received the highest rated dimension for both practices. All subjects had fun with the pair or group and reported that the practice promoted cooperation between the students.

4.3. Learning

The majority of the students also expressed that they believe that both practices contributed positively to their long-term learning (Figure 3, see the left side for Pair Programming and right side for Coding Dojo), indicating that the experience with practice could be useful in development mockups in working life. The main aspect cited by the students was to know how to work in a team, subject 1 from Pair Programming group said: "Pair Programming helps to learn to work as a team to get better performance." Subject 3 from the Coding Dojo group reported that the practice will help in the future use of pair programming: "With Coding Dojo I will know better how to behave in case of a pair programming."

The learning was also confirmed by the student's responses with respect to the perceived impact on the knowledge levels in accordance to Bloom's taxonomy [21]. In this taxonomy, the students reported in a range varying from 1-5 their perceived knowledge in the concepts before and after the use of the practice. The students from both groups (Figure 4 for Pair Programming and Figure 5 for Coding Dojo) perceived a significant increase of knowledge with respect to all three concepts taught: mockups, interaction between mockups and Qt syntax by the practice on all three knowledge levels.

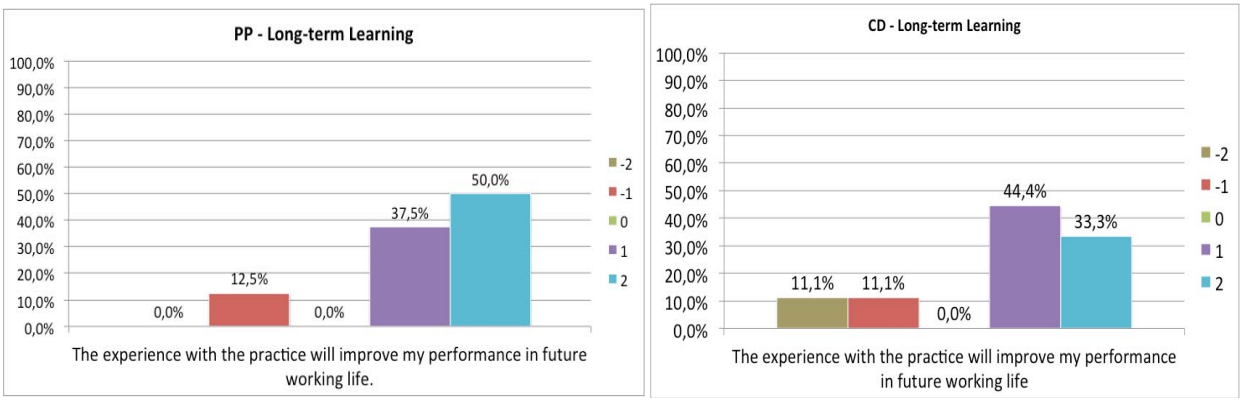

Figure 3. Frequency diagram about the item about long-term learning in both groups

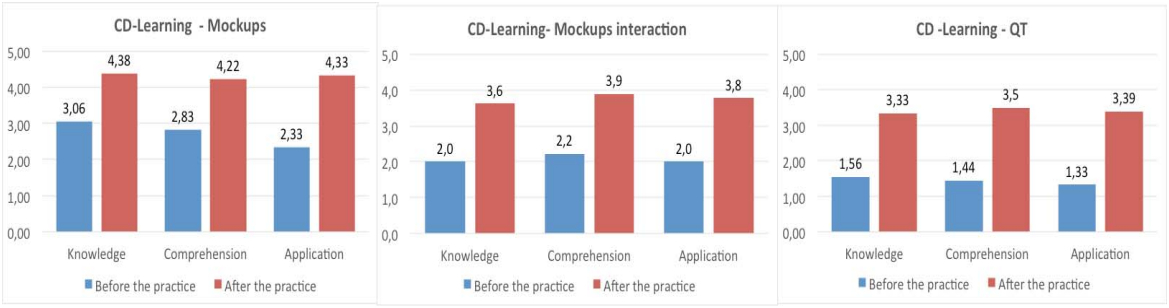

Figure 4. Grades of the Learning dimension with Bloom's taxonomy in Pair Group [21]

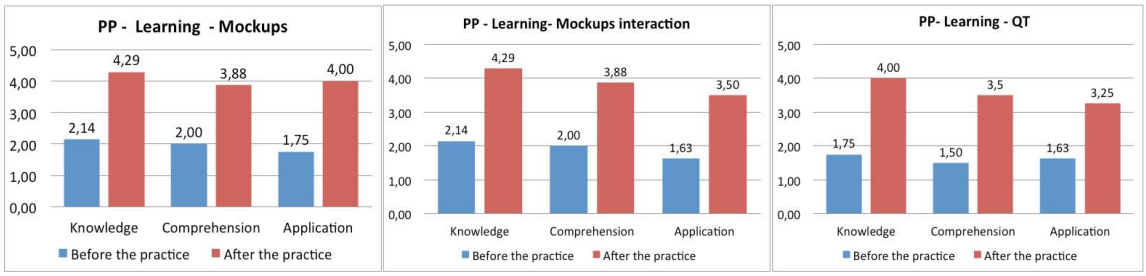

Figure 5. Grades of the Learning dimension with Bloom's taxonomy in Coding Dojo Group [21]

4.4. Benefits

The open-ended questions helped us to identify the following benefits.

4.4.1. Pair Programming. The students that used pair programming reaffirmed the social interaction and knowledge transfer of the practice. Subject 1 reported: *"The interaction with the pair is crucial to practice to be successfully developed, which increases learning."* Subject 4 reinforce this, saying: *"The interaction with the pair facilitates learning."*

Another benefit of Pair Programming reported by the students is about the creation of a programming style. According to the students, Pair Programming helps the team to create a solution that could be common to the both students. Subject 6 reported: *"The practice leads us to create patterns to be understood by both students."*

4.4.2. Coding Dojo. Regarding the benefits of Coding Dojo, the students reported the detection of defects. Subject 3 said: *"Other people can find the defect quickly who programmed."*

Learning was also reported as benefit, as Subject 3 said: *"In Coding Dojo, there is more chance to learn programming techniques."*

The students reinforce the fun in Coding Dojo. Subject 2 reported: *"Coding Dojo promotes a greater non-competitive among the participants, making the programming practice a little more fun. "*

4.5. Disadvantage

The open-ended questions helped us to also identify the following disadvantages.

4.5.1. Pair Programming. The lack of consensus was the highest cited disadvantage of Pair Programing, as perceived by the students. Subject 1 reported: *"If the pair does not agree on how to do, may have a delayed delivery time and also affect the quality of the product."*

The infrastructure was reported as essential to the practice. Subject 4 said: *"Very quiet environment, the practice requires a greater communication"*. The students reported other negative points such as the difference of knowledge between the pairs. Subject 8 said: *"If one of the pair does not have sufficient knowledge of the content of the session, may contribute less. "*.

The time of the study and also the moment of switching roles were cited as drawbacks in the group of Pair Programing. Subject 1 said: *"When there is an interruption to switch the pairs, our concentration can be lost sometimes, which prejudice the programming."*

4.5.2. Coding Dojo. Regarding Coding Dojo, the main disadvantage cited by the students was the goal conflict between the pairs, and the lack of consensus. Subject 1 said: *"Pair Programming is not difficult to understand or practice, but it's difficult to accept the ideas of the group at all."*

Subject 2 said: *"In several moments, I tried to convince other people to do as I do because I judged to be better, even without seeing the result of what was being done."*

The students also cited the short duration of time box, the dispersion of students between the audience and the infrastructure (due a bad visualization of the code for all the audience).

4.6. Mockups

We have distributed five mockups to be developed by the students. However, no one has finished all of them. In the Pair Programming group, only one of the four pairs developed two mockups, the other three pairs finished the first and started to develop the second mockup. In addition, only two pairs running the mockups in Qt, the other two pairs presented mockups with compilation error.

In the Coding Dojo group, the students developed only one mockup. During seven sessions the students tried to fix an error, and only in the last 3 sessions they finally achieved the solution.

5. Threats to Validity

One of the key issues in empirical studies is evaluating the validity of the results. In this section we discuss the potential threats that are relevant for our study and how they are addressed.

5.1. Construct Validity

According to Wohlin et al. [19] the construct validity is concerned with the relationship between the theory and the observation. In this study to evaluate the learning, motivation and user experience, we have followed the construct proposed by Wangenheim et al. [20].

5.2. Internal Validity

Threats to internal validity have influence in the conclusions about a possible causal relationship between the treatment and the outcome of a study. In this study we considered four main threats to the internal validity: (a) training effects, (b) subjects' programming and Qt expertise, (c) pair programming and coding dojo expertise, and (d) type of the tasks.

In relation of the training effect, there could be a risk if the quality of the training had been different in from one group to the other. However, we controlled this threat by giving a similar training for both groups.

Furthermore, in order to mitigate the threat of the subject's programming knowledge, we divided them into balanced groups according to their experience. This measure avoided that the subjects' experience affected the overall results of the practices. Another problem could have been the expertise in pair programming and coding dojo, we also tried to balance the group, merging this skill with programming experience.

Regarding the type of tasks, we controlled this threat in both practice, using the same set of mockups to develop. Both group also receive the same Qt settings and material.

5.3. External Validity

External validity describes the study representativeness and the ability to generalize the results outside the scope of the study [19] Each University has different approaches to teach developments mockups in different levels and periods of a graduate course. The language or tool to develop a mockup may vary. In addition, we cannot generalize the results in environments outside the academy, such as in industry training, for instance. In

5.4. Conclusion Validity

The conclusion validity is concerned with the relationship between the treatment and the results [19]. In this study, the biggest problem is the small number of subjects and it was only possible to create one Coding Dojo group._Other threat identified is that all the students in the study came from the same University. For this reason, the data extracted from this study presents important results related to motivation, user experience and learning, but can not be generalized at this time. More studies and replications are needed in the future.

6. Discussion

Pair Programming and Coding Dojo represents an attempt to teach front-end programming tasks such as the development of mockups, providing an environment with motivation among all the students involved, a great user experience and learning.

The feedback obtained provides evidence that the two practices could be effective in the learning of mockups development. As collaborative practices, both Pair Programming and Coding Dojo showed the need of a specific infrastructure that allows the students to use the practice effectively. For instance, in Pair Programming a room to support open communication is necessary, while in Coding Dojo it is necessary a good visualization of the code by the audience.

Both practices presented challenges related to creating a consensus between the students. We believe that a deep investigation in the personality types of the students could explain this context, as Salleh et al. [8] pointed out, but we did not collect enough data to support this claim. On the other hand, Pair Programming and Coding Dojo presented positive results related to social interactions, cooperation and knowledge transfer. In Coding Dojo, as the mockups were developed within a group, the consensus was more difficult in each time box.

The mockups developed by the two pairs in the Pair Programming group had more requirements quality than the mockup developed by the Coding Dojo group. On the other hand, the code delivered by the Coding Dojo group presented a better code quality.

Additionally, in the Pair Programming group, two pairs delivered mockups with errors (without executing in the Qt tool). However, in the Coding Dojo group we noticed the need of more time to develop a solution by the students. In the Pair Programming group, the students seemed to be more focused in developing the solution, but there was a lack of care in the code and in the development (only two pairs delivered mockups without errors).

The students were more interested by Pair Programming than Coding Dojo. This can be explained by the dispersion level between the pairs when compared with the group in Coding Dojo. However, more investigation is need in this topic. The literature, for example, cites Pair Programming as an established practice with an impact in long-term learning [8].

7. Conclusions and Future work

In this paper, we present an empirical study where we investigated the use of Pair Programming and Coding Dojo in the development of mockups. This study was planned and executed within an Analysis and Design course and the main goal was to improve the learning process of the students. Overall, Pair Programming and Coding Dojo presented positive results in the learning process, but Pair Programming had better results than Coding Dojo, specifically in terms of learning and user experience. Both practices are easy to understand and use, having positive results in relation to fun and user experience. Coding Dojo showed to be more challenging in respect to a consensus about the session goal.

Future steps in this work involve the planning and execution of new empirical studies in order to evaluate Pair Programming and Coding Dojo in other types of tasks. We expect that our findings could be useful for higher education professors and students, providing an overview of the use of both practices in the teaching of mockup development. We also hope that this work could help practitioners in the teaching of these subjects in the context of the software industry.

8. Acknowledgements

This research is partially funded by the National Science Foundation (grant 1242257, Pan American Software Quality Institute). We would also like to thank CNPq (309000/2012-2), FAPERGS, FAPEAM (062.00146/2012; 062.00600/2014; 062.00578/2014; and 01135/2011), and the research agreement between PUCRS and ThoughtWorks.

9. References

[1] T. Inoue, “Relation between Behavior and Result in Pair Programming: Talk and Work Leads to Success,” In: Proceedings of the 21st International Conference on Computers in Education. Indonesia: Asia-Pacific Society for Computers in Education (Nara, Japan), pp. 275-280, 2013.

[2] R. Arora and S. Goel., "Learning to Write Programs with Others: Collaborative Quadruple Programming," IEEE 25th Conference on Software Engineering Education and Training (CSEE&T), (Nanjing, China), pp.32-41, IEEE, 2012.

[3] K. Beck and C. Andres, “Extreme Programming Explained: Embrace Change”. Addison-Wesley Professional , 2 .ed, 224p, 2004.

[4] C. Mcdowell, L. Werner, H. Bulock and J. Fernald, “The effects of pair-programming on performance in an introductory programming course,” SIGCSE technical Symposium on Computer Science Education, (Cincinnati, USA), pp. 38–42, ACM, 2002.

[5] N. Ramli and S. Fauzi, “The effects of pair programming in programming language subject,” International Symposium on Information Technology, (Kuala Lumpu, Malaysia), pp.1-4, IEEE, 2008.

[6] E. Wiebe, L. Williams, J. Petlick, S. Balik, C. Miller and M. Ferzli, “Pair programming in introductory programming labs,” American Society for Engineering Education Annual Conference & Exposition, (Nashville, USA), 12p., 2005.

[7] B. Simon and B. Hanks, “First-year students’ impressions of pair programming in CS1”.J. Educ. Resour. Comput., vol. 7-4, pp. 1-20, ACM, 2008.

[8] N. Salleh, E. Mendes and J. Grundy, “Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review,” IEEE Transactions on Software Engineering, vol.37-4, pp. 509–525, 2011.

[9] A. Cockburn and L. Williams, "The costs and benefits of pair programming," Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, pp. 223-243, 2001.

[10] D.T Sato, H. Corbucci and M.V Bravo, "Coding Dojo: An Environment for Learning and Sharing Agile Practices," Agile Conference, (Toronto, Canada), pp.459-464, IEEE, 2008.

[11] K. Heinonen, K. Hirvikoski, Matti Luukkainen, and Arto Vihavainen, "Learning agile software engineering practices using coding dojo," Proceedings of the 14th annual Conference on Information Technology Education (SIGITE '13), (New York, USA), pp. 97-102, 2013.

[12] R.B. Da Luz, A.G Neto and R. Noronha, "Teaching TDD, the Coding Dojo Style," International Conference Advanced Learning Technologies (ICALT), (Beijing, China), pp.371-375, IEEE, 2013.

[13] G. Canfora, A. Cimitile, F. Garcia and M. Piattini, C.A. Visaggio, "Performances of pair designing on software evolution: a controlled experiment," European Conference on Software Maintenance and Reengineering, (Bari, Italy) ,pp.8, IEEE, 2006.

[14] K. Lui, K.C.C Chan and J.T. Nosek, "The Effect of Pairs in Program Design Tasks," IEEE Transactions on Software Engineering, vol.34, no.2, pp.197-211, 2008.

[15] E.R. Luna, J. I. Panach, J. Grigera, G. Rossi and O. Pastor, "Incorporating usability requirements in a test/model-driven web engineering approach," Journal of Web Engineering, vol. 9 , pp.132-156, IEEE, 2010.

[16] N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Miller and S. Balik, "Improving the CS1 experience with pair programming," SIGCSE technical Symposium on Computer Science Education , (Reno, USA), pp. 359-362, ACM, 2003.

[17] J. Vanhanen, C. Lassenius, and M.V. Mantyla, "Issues and Tactics when Adopting Pair Programming: A Longitudinal Case Study," Software Engineering Advances, (Cap Esterel , France), pp. 25-31, IEEE, 2007.

[18] M. Müller, "Do programmer pairs make different mistakes than solo programmers?,". Journal of Systems and Software, vol.80-9, pp.1460–1471, 2007.

[19] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, and A. Wessl, "Experimentation in software engineering: an introduction, Kluwer Academic Publishers, 2000.

[20] C. Wangenheim, R. Savi, R. Borgatto, "SCRUMIA- An educational game for teaching SCRUM in computing courses," Journal of Systems and Software, vol. 86 - 10, pp. 2675-2687, 2013.

[21] B.S. Bloom, "Taxonomy of Educational Objectives: The Classification of Educational Goals,": Handbook I. Cognitive Domain, Longmans, 1956.