Does Pair Programming work in a Data Science Context? An Initial Case Study

Jeffrey S. Saltz
Syracuse University
Syracuse, NY
jsaltz@syr.edu

Ivan Shamshurin
Syracuse University
Syracuse, NY
ishamshu@syr.edu

Abstract— While pair programming has been studied extensively for software programmers, very little has been reported with respect to pair programming in a data science project. This paper reports on a case study evaluating the effectiveness of pair programming within a data science / big data context. Our findings show that pair programming can be useful for data science teams. In addition, while the *driver* role was similar to what has been described for software programmers, we note that the *observer* role had an expanded set of responsibilities, which we termed *researcher* activities. Further exploration is required to explore if these expanded roles are specific to data science pair programming.

Keywords- Big Data, Data Science, Project Management, Pair Programming, Process Methodology

I. INTRODUCTION

Pair programming (PP) is a practice of code development when two programmers work side-by-side at one computer. The first person, the driver, has a full control of the keyboard and mouse, and implements the code. The other programmer, the observer or navigator, continuously observes the work of the driver to identify tactical (syntactic, spelling, etc.) defects, and also thinks strategically about the direction of the work [1]. Similar to a co-driving in a rally, navigators also consider the "strategic" direction of the work, coming up with ideas for improvement and addressing likely future problems. This allows driver to focus on the tactical aspects of the task.

Pair programming has been used in software engineering for decades. However, using this approach in big data/data science and other relevant domains is limited. In contrast to software engineering (SE), pair programming has not been widely adopted in big data, data science and other domains. The goal of this study is to address this gap by answering the following research questions:

RQ1. Is PP helpful for big data/data science projects?

RQ2. What are the differences in using PP for SE vs big data/data science projects?

The rest of the paper is organized as follows. First, a literature review on using PP in SE, big data/data science and other domains will be described. Then an in-class experiment on using PP within a data science course will be presented. Lastly, a synthesis of our observations will be provided within our discussion and conclusion.

II. BACKGROUND

A. *Previous research using Pair programming in Data Science*

A systematic literature review (SLR) is a way of identifying, evaluating and interpreting research relevant to a particular research question, topic area, or phenomenon of interest, using a research method that is reliable, accurate and facilitates auditing [2]. Our efforts to summarize existing research and to identify gaps in current research with respect to the process and methodologies teams use to execute Big Data projects followed the SLR guidelines outlined by Kitchenham [3].

Specifically, a Google Scholar search was employed to get relevant papers. Table 1 shows the two different Google Scholar searches that were used to identify relevant research papers. The search terms included "big data", "data science" or "analytics" as well as "pair programming". The top 100 search results for each of the queries below were analyzed.

Table 1: Search Terms Used

Source	Search Terms
Google	"pair programming" AND ("data science" OR "analytics" OR "big data")
Google	"pair programming data science", "pair programming analytics", "pair programming big data"

We stopped at 100 articles due to our lack of relevant articles identified. Specifically, no relevant academic papers were found. However, it was noted that PP was used in Data Science classes at top schools, ex. NYU data science initiative [4]. In addition, more recently, some pair programming initiatives have been started, ex. PP for Kaggle competitions [5, 6].

B. *Potential Applicability of Pair programming in Data Science*

While there are many roles required to do data science effort [7], many of those roles require significant programming, and hence, pair programming within those roles is likely as beneficial as pair programming has been shown to be useful for software development. In addition, it has been noted that the benefit of pairing is greatest on tasks that the programmers do not fully understand before they begin, such as challenging tasks that requires creativity and sophistication [8]. This description of the task is a typical case for big data/data science projects, because these projects are exploratory in nature [9].

In addition, pair programming is part of a range of activities related to agile development. While the concept of agile development within data science is not well understood, initial studies suggest that some forms of agile development are appropriate for data science teams [10]. For example, communication is crucial for Big Data/Data Science projects and it has been reported that three quarters of corporate business intelligence projects fail due to poor communication [11]. As previously noted, pair programming helps programmers communicate more easily [12]. Furthermore, pair programming has proven to be a useful approach to facilitate effective knowledge sharing [13]. Hence, by adopting pair programming, data science teams might be able to overcome communication issues and work in a more agile manner.

Furthermore, a meta-analysis of 15 years of empirical pair programming tests [14] has shown that pairs generally learn more. So, by leveraging the learning aspects of pair programming, pair programming could help junior data scientists quickly learn from their senior colleagues. Finally, pair programming creates a boundary condition (such as between a team member trying to solve the challenge quickly and a team member who wants to move in a specific exploratory direction), and boundary conditions have been shown to improve learning outcomes [15] and hence, team performance.

C. *Pair programming in Software Engineering*

While little has been explored with respect to data science teams using pair programming, pair programming has been extensively studied within a software development context. There are both empirical studies and meta-analyses of pair programming for software development. The empirical studies tend to examine the level of productivity and the quality of the code, while meta-analyses may focus on biases introduced by the process of testing and publishing.

Within these studies, it has been noted that software engineering is similar to team sport, where the team's work will impact the project's quality, the team's morale, and their ability to solve complicated tasks [16]. With respect to teamwork, pair programming has multiple advantages and disadvantages over traditional non-pair programming. We briefly summarize key findings such as code quality, level of effort, programmer satisfaction and learning, and team building and communication.

Code Quality

It has been shown that despite an increase in the man-hours required, the resulting code has about 15% fewer defects [12]. Thus, pair programming significantly reduces these expenses by reducing the defects in the code and resources required for code maintenance [12]. In a similar meta-analysis study [14] it was shown that pairs generally produce higher quality code.

In addition, typically two programmers working side-by-side generate a larger number of solutions to problems than a single programmer. The two programmers have to negotiate a shared direction of work and find solutions to conflicts between them when they arise. In doing so, these pairs consider a larger number of ways to solve the problem. This significantly improves the design quality of the program as it reduces the chances of selecting a poor method [17]. The solutions are more diverse because, for example, the two people might have different backgrounds and experiences and therefore have different perspectives on the same task.

Level of Effort

Pair programming typically increases the time required to deliver code compared to programmers working individually. According to several studies, this increase varies from 15% to 100% [8]. However, testing and debugging time might decrease and managers have to balance faster completion of the work and reduced testing and debugging time against the higher cost of coding. The relative weight of these factors can vary by project and task.

In contrast to increase in code development time, other factors such as support cost and quality assurance may decrease in cost. That is important because usually code support is a significant percent in the product lifecycle. For example, IBM reported spending about "$250 million repairing and reinstalling fixes to 30,000 customer-reported problems" [18].

However, in some cases pair programming may result in a net drop in productivity [8, 19]. For example, teams usually get low productivity when they work on simple tasks using PP. The pair already fully understands the task and therefore there is no need to use PP. Another example of a productivity drop is when a novice–novice pairing is used without sufficient mentoring to supervise the pair [20].

Programmer Satisfaction & Learning

In an online survey of pair programmers [21], 96% of programmers stated that they enjoyed their work more than when they programmed alone and 95% said that they were more confident in their solutions when they pair programmed.

In industry or in the classroom, pair programming is an efficient technique for learning because knowledge is constantly shared between pair programmers. Pair programming helps novices to quickly learn from their more experience peers and several studies suggest that students show higher confidence when programming in pairs [21]. This learning might cover both tips on programming language and overall design skill [22]. In other words, pair programming reduces the skills and experience gap among team members.

Pair programming develops critical thinking of navigators/observers because it allows the programmers to examine their partner's code and provide feedback. By doing this, these programmers develop monitoring mechanisms for their own learning activities [22].

Team building and communication

Communication is a key component in SE, data science and many other domains such as business intelligence [11]. Pair programming allows team members to share problems and solutions quickly, making the two people to be less likely to have hidden agendas. This helps pair programmers learn how to more easily communicate, in other words, "this raises the communication bandwidth and frequency within the project, increasing overall information flow within the team" [12].

III. Methodology

Pair programming within a data science context was explored via a case study. Merriam [23] indicated that with case study research, it is important to have a bounded system that can be identified as the focus of the investigation. This case study focused on students within a graduate level applied data science course using pair programming for data science tasks. The rest of this section describes the case study in more detail by first describing the subjects in the study and the context of the case study. We then describe a model that we defined to evaluate and compare the use of pair programming, and finally, we report on the findings from the experiment.

A. Background / Context

Pair programming was evaluated within a graduate-level introduction to data science course. A total of 110 graduate students participated in the study. Five sections participated in the study, with 20 to 24 students per section. The students had diverse geographic / cultural backgrounds, with students gaining their undergraduate degree from Asia, Europe and North America. In fact, while all the students attended the course in a face-to-face format in North America, the majority of students had previously been educated outside of the United States. In addition, forty percent of the students were female and more than 75% of the students had previous IT experience. A majority of the students had two to five years of work experience, typically within the IT industry. This experience was gained after they earned their undergraduate degree. For half the students, this was their first exposure to data science. The other half of the students had some basic introductory exposure to the field of data science.

B. The Use of Students

While there has been little documented on data science case studies and experiments, students have typically been used in software development experiments. For example, it was observed that students were used as subjects in 87% of the experiments analyzed over a ten year period [3]. However, it has been noted that "students vs. professionals" is actually a misrepresentation of the confounding effect of proficiency, and in fact differences in performance are much more important than differences in status [24]. Hence, using master level students, with an average of 3 years IT experience can often be a more appropriate choice than undergraduate students with minimal experience. In addition, while students might not be as experienced as practicing professionals, they can be viewed as the next generation of professionals and are suitable subjects for many software development experiments [25, 26].

In addition, it was noted that "in the academic world of computer programming, little real world is taking place" [27] because universities teach solitary programming and non-incremental development. Using pair programming in class addresses this gap because it turns learning into team, incremental and iterative activity [27].

C. Case Study Context / Environment

In addition to the class's weekly lecture, each section also met weekly at a specific day and time for a 90 minute lab session. For part of the lab session, students were required to work on a data science assignment. Each week there was a different assignment. For most of the assignments, students were required to use the R programming language, a popular data science tool that is used in both industry and academia. For these assignments, the student teams were expected to do R programming, using typical data science techniques such as machine learning algorithms and geographic information analysis.

In terms of the assignments, while big data challenges are typically described in terms of the four Vs (volume, variety, velocity and veracity), a more useful description of the assignment, in this context, is based on two key dimensions of a project, the level of discovery needed and the project's infrastructure requirements [28]. The assignments did not require big data infrastructure, but did require more open-ended hypothesis generation analysis.

For the first four weeks, students worked on the assignments individually. After that, students were instructed on the benefits of pair programming as well as how to do pair programming within the data science context. Students were randomly assigned into teams of two people, with the two-person teams being held constant throughout the semester.

Three different instructors taught the five sections. All of the instructors had previously taught the data science course several times and were knowledgeable with respect to the course content and the tools available within the online learning environment.

D. Pair Programming Process

For the first four sessions, the process used within the lab session was that the students worked on their own. This baseline condition is the current practice of most typically used by data scientists (and data science students).

Before the first use of pair programming, the process was explained to students via discussions and a documented presentation. Specifically, pair programming consisted of the following steps. First, there was one *driver* (the person that had control of the shared screen, and was typing within a shared document). The other student was the active *observer* that, via a shared screen, reviewed what the driver was writing. The observer read what the driver was writing, and thought about larger issues, such as what task to do next and issues with what was currently being written. The observers were instructed to not dictate to the drivers what should be written. All the students were encouraged to be actively engaged with each other, for example, sharing their thoughts and ideas and asking questions. Finally, students were instructed to frequently rotate which student was the driver, with a goal of rotating every fifteen minutes.

E. *Model to Evaluate Team Performance*

To compare different methodologies (using pair programming or not using pair programming), it is necessary to evaluate and compare the effectiveness of the different teams [10]. Thus it is important to define a theoretical model that can aid in the evaluation of the different methodologies. To measure the effectiveness of the methodologies, we integrated two models, as shown in Figure 1. First, we leveraged Ko et al's [29] systematic review of over 1,700 software engineering papers focusing on tool evaluations. This review identified a common approach of measuring success on task, including ***solution quality***, and ***user perceptions*** such as perceived usefulness and perceived ease of use [29]. Second, we leveraged Hackman's model [30], which is one of the most widely used normative models for identifying factors related to team effectiveness. In brief, this model focuses on the input factors (such as organizational context and group design), process and moderating factors and the output factors (including task output, the team's continued capability to work together and the satisfaction of individual team members). Since we controlled the input, process and moderating factors, in our model, our focus was on ***task output*** (similar as Ko's solution quality) and ***satisfaction with team*** (which includes the team's continued capability to work together and the satisfaction of individual team members). Note that our focus on ease of use includes both the student perspective (is the methodology easy to use) and the instructor perspective (how much time does the methodology require, scalability to support many teams).

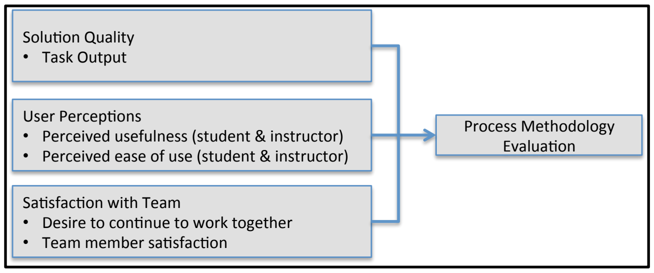
Solution Quality
• Task Output
User Perceptions
• Perceived usefulness (student & instructor)
• Perceived ease of use (student & instructor)
Satisfaction with Team
• Desire to continue to work together
• Team member satisfaction
Process Methodology Evaluation

Figure 1: Process Evaluation Model

F. *Data Sources*

In this case study, multiple data sources were used, which is consistent with Eisenhardt [31]. First, during each lab session, the instructor observed the student teams doing pair programming and documented those observations via answering semi-structured questions. The observations focused on the perceived satisfaction and productivity of the team members while using pair programming. This provided an initial view into task output (via the perceived productivity of the team members) as well as the satisfaction of the team members (via the instructor's observations of how the team members interacted).

In addition, student surveys, which included 5-level Likert questions as well as more open-ended questions, were used to collect student perceptions on their use of pair programming. These student questionnaires asked several structured questions, which provided quantitative data on topics such as would the student like to work with their other team members on future projects, how well the team worked together and did they find the methodology easy to use. The survey also had semi-structured questions that focused on what worked well for each team.

Finally, data was also collected from project artifacts, such as the graded assignments. The qualitative data was analyzed via an iterative process of item surfacing, refinement and regrouping. Table 2 maps the data sources to the key measures defined in our model to evaluate the effectiveness of pair programming.

Table 2: Measuring Team Effectiveness

Key Measures	How Measured
Solution Quality / task output	Grading of tasks, Instructor observations
User Perceptions – Perceived usefulness	Student survey, Instructor survey
User Perceptions – Perceived ease of use	Student survey, Instructor survey
Satisfaction with team	Student survey
Satisfaction with team - Satisfaction of team members	Student survey

IV. FINDINGS

In this section, based on our model to evaluate pair programming, described in Figure 1 and the results from our and student and instructor surveys, we report on our findings. Specifically, for the student survey, 95 of the 110 students responded to our end of semester survey, and the key results are highlighted in Table 3. These results are discussed below.

Table 3: Student Perceptions of Pair Programming

	Mean	% Agree
PP improves productivity	4.4	95%
PP was useful	4.4	91%
PP was easy to use	4.5	92%
I want to work with my partner again	4.4	90%
I want to use PP for future assignments	4.3	85%

A. Solution Quality

Task output was evaluated via reviewing the grades of the weekly assignments. The goal was to determine if the grades were higher when using pair programming (as compared to the initial four weeks, when students worked individually). However, this was not the case. In fact, there was no material difference in the grades across the two conditions (the first four weeks vs the rest of the semester). Upon reflection and follow-up discussions with students, it was determined that this was due to the fact that the assignments did not have to be turned in at the end of the lab session. Rather, students had the option of continuing to work on the assignment for several more days. Hence, students were able to put in more time to improve their assignments if needed. This was addressed by adding a question at the end semester student survey, asking if pair programming improved their productivity. With respect to this question, as shown in Table 3, 95% of the students agreed or strongly agreed that 'PP improves our productivity'. Overall, the average response was 4.4 (on a 5-point Likert scale).

B. Percieved Usefulness and Ease of Use

Students thought that pair programming was useful and easy to use. Specifically, as shown in Table 3, 91% of the students agreed or strongly agreed that 'PP was useful' and 92% of the students thought that 'PP was easy to use'.

The open-ended student qualitative feedback was used to more deeply explore the usefulness. Two key themes emerged, each of which are described below.

Improved Quality of their Design – students noted that they thought that they had improved quality. For example, one student noted that when using PP "two people will have two different methods to solve the problem. So working together we can learn about what the other person thinks. We can learn another method to solve the same question". A similar thought was expressed a bit differently by another student "was very helpful as we could brainstorm and think of more accurate answers".

Improved Speed – students thought that they worked faster when using PP. Note, however, that students might have been comparing to their baseline case of working by themselves. In any event, in exploring the students' qualitative answers, the speed improvements were clear. Sometimes students were very direct, such as "helps to complete the assignment much faster". But sometimes students explained their logic, such as "it is easy for the observer to spot small mistakes, which could have been overlooked by driver". This contrasts to typical findings in the software development domain, where several studies have shown that coding time typically increases [8]. This is likely due to the fact that students were not just thinking of coding time, but rather, time to design, code and debug the data analysis.

C. Satisfaction with Team

As can be seen in Table 3, students were very satisfied when using pair programming. Specifically, 90% of the students agreed (or strongly agreed) that they 'want to work with my partner again', and 85% of the students 'want to use PP for future assignments'. For example, one student noted that "Teamwork ability, communication, and cooperation skills are also improved by pair programming". A key theme that emerged from the students' open-ended responses was that they were able to learn from their partner. For example, one student noted "my partner helped me figure out things that I would have otherwise been stuck on for longer, and I did the same for her". Another person observed "as someone who has little experience in data science and information science, I always find it useful to see how others work, and this was a great way to do that".

D. Instructor Observations / Feedback

Overall, the instructors received positive feedback on the use of pair programming and were supportive of continuing to use pair programming with future teams.

One interesting theme that emerged from the instructors was that many teams were using a new model of pair programming, this new model, which we call *driver-researcher,* expanded the role of the observer. In addition to actively watching the driver, the observer (who we now call researcher) was actively exploring implementation questions (ex. exploring an appropriate model or how to best visualize the data). As one instructor noted, "sometimes the 'observer' is not just observing, but doing research (ex. googling) while the second one was coding". Another instructor noted that "this new model is interesting - and I have seen it a lot".

V. Conclusion

This case study explored the use of pair programming in a data science context. While much has been explored with respect to pair programming for software development, very little has been reported with respect to pair programming for teams doing data science activities. Our case study focused on R pair programming data science activities. Perhaps not surprisingly, we found pair programming to have similar benefits as has been found within a software development context. In particular, our case study revealed that pair programming is effective for data scientists to improve communication and create better code in less time. In addition, similar to software development classes [27], in an applied data science class, pair programming also had a perceived significant positive effect on student's task outcomes. Hence, this research answers the first research question by noting that pair programming is helpful for big data/data science projects.

One interesting finding that we observed was a new set of tasks that were done by the *observer*. Hence, we have a view on our second research question (what are the differences in using PP for SE vs big data/data science projects). However, more work is required to fully explore this observation. Specifically, future research should explore if these new responsibilities for the observer are only appropriate within a data science context, or are they are also applicable for software development programmers.

One limitation within this case study was that all the data scientists studied were students. It is possible that there are different challenges when pair programming is used by full-time data scientists in industry. Another limitation is that the pairs of people were randomly assigned. It might be interesting to explore matching novices together (or novices with more advanced data science knowledge/experience). In particular, it has been found that there are differences in the patterns between different pair combinations [32].

A possible next step might be to explore the accuracy and volume of work generated by each individual. For example, study the impact on the 'volume per unit time' productivity of the individuals.

Finally, one might explore remote pair programming to see if remote pair programming is as efficient as face-to-face data science pair programming. This study will be important for online data science classes as well as for data science teams in industry, where teams are often globally distributed.

VI. REFERENCES

[1] Williams, L. (2001) Integrating Pair Programming Into a Software Development Process, 14th Conference on Software Engineering Education and Training. Charlotte. pp. 27–36. ISBN 0-7695-1059-0. DOI:10.1109/CSEE.2001.913816

[2] Kitchenham, B. (2004). Procedures for performing systematic reviews. Keele, UK, Keele University, 33(TR/SE-0401), 28. http://doi.org/10.1.1.122.3308

[3] Kitchenham, B., Charters, S. (2007). Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3. Engineering, 45(4ve), 1051. http://doi.org/10.1145/1134285.1134500

[4] NYU (2017) https://nyu-cds.github.io/courses/programming/

[5] Kaggle1 (2017) https://www.meetup.com/fr-FR/Greater-Cleveland-SciPy-Julia-R-Data-Science-Group/events/234024079/

[6] Kaggle2 (2017) https://www.meetup.com/en-AU/Greater-Cleveland-SciPy-Julia-R-Data-Science-Group/events/241583050/

[7] Saltz, J. & Grady, N. (2017). The Ambiguity of Data Science Team Roles and the Need for a Data Science Workforce Framework, In *Big Data (Big Data), IEEE International Conference on*. IEEE.

[8] Lui, K. M. (2006) Pair programming productivity: Novice-novice vs. expert-expert. International Journal of Human–Computer Studies. 64 (9): 915-925. DOI:10.1016/j.ijhcs.2006.04.010

[9] Saltz, J. (2015) The Need for New Processes, Methodologies and Tools to Support Big Data Teams and Improve Big Data Project Effectiveness, In *Big Data (Big Data), IEEE International Conference on*. IEEE

[10] Saltz, J., Shamshurin, I., Crowston, K. (2017). Comparing data science project management methodologies via a controlled experiment. Hawai'i International Conference on System Sciences (HICSS-50)

[11] Jensen, P. (2004). What is Operations Research? DOI= http://www.me.utexas.edu/~jensen/ORMM/models/unit/or_method/process.html

[12] Cockburn, A., Williams, L. (2000) The Costs and Benefits of Pair Programming. Proceedings of the First International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2000)

[13] Kavitha R., Irfan Ahmed, M. (2013) Knowledge sharing through pair programming in learning environments: An empirical study, Education and Information Technologies, June 2015, vol. 20, Issue 2, pp 319–333

[14] Salge CA de Lima, Berente, N. (2016) Pair Programming vs. Solo Programming: What Do We Know After 15 Years of Research? 49th Hawaii International Conference on System Sciences (HICSS), DOI: 10.1109/HICSS.2016.667

[15] Heckman, R., Østerlund, C. S., & Saltz, J. (2015). Blended learning at the boundary: Designing a new internship. Online Learning, 19(3), 1-17.

[16] Fitzpatrick B., Collins-Sussman B. (2012). Team Geek: A Software Developer's Guide to Working Well with Others, O'Reilly Media, Sebastopol, CA.

[17] Flor, N. V.; Hutchins, E. L. (1991). Analyzing Distributed Cognition in Software Teams: A Case Study of Team Programming During Perfective Software Maintenance". In Koenemann-Belliveau, J.; Moher, T. G.; Robertson, S. P. Empirical Studies of Programmers: Fourth Workshop. Ablex. pp. 36–64. ISBN 978-0-89391-856-9

[18] Humphrey, W.S. (1995) A Discipline for Software Engineering. SEI Series in Software Engineering, ed. P. Freeman, Musa, John. 1995: Addison Wesley Longman, Inc.

[19] Arisholm, E., Hans Gallis; Tore Dybå; Dag I.K. Sjøberg (2007). Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. IEEE Transactions on Software Engineering. 33 (2): 65–86. doi:10.1109/TSE.2007.17

[20] Stephens, M., D. Rosenberg (2003) "Will Pair Programming Really Improve Your Project?, Methods and Tools

[21] Williams, Laurie; Kessler, Robert R.; Cunningham, Ward; Jeffries, Ron (2000). "Strengthening the case for pair programming". IEEE Software. 17 (4): 19–25. doi:10.1109/52.854064.

[22] Williams, Laurie; Upchurch, Richard L. (2001). In support of student pair-programming. ACM SIGCSE Bulletin. 33 (1): 327–31. doi:10.1145/366413.364614.

[23] Merriam, S. (1998). *Qualitative research and case study applications in education*. San Francisco: Jossey-Bass

[24] Soh Z., Sharafi Z., Van den Plas B., Porras G. C., Guéhéneuc Y.-G., Antoniol G. (2012) Professional status and expertise for UML class diagram comprehension: An empirical study, in *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on*, 2012, pp. 163-172: IEEE.

[25] Kitchenham B.A., Pfleeger S.L., Pickard L.M., Jones P.W., Hoaglin D.C., Emam K. E., Rosenberg J. (2002) Preliminary guidelines for empirical research in software engineering, *Software Engineering, IEEE Transactions on,* vol. 28, no. 8, pp. 721-734, 2002.

[26] Salman I., Misirli A. T., Juristo N. (2015) Are students representatives of professionals in software engineering experiments?, in *Proceedings of the 37th International Conference on Software Engineering-Volume 1*, 2015, pp. 666-676: IEEE Press.

[27] Umapathy K., Ritzhaupt A.(2017) A Meta-Analysis of Pair-Programming in Computer Programming Courses: Implications for Educational Practice, *ACM Transactions on Computing Education (TOCE)*, vol. 17 Issue 4, September 2017

[28] Saltz, J., Shamshurin, I., Connors C. (2017) Predicting data science sociotechnical execution challenges by categorizing data science projects. *Journal of the Association for Information Science and Technology*, DOI: 10.1002/asi.23873

[29] Ko A., LaToza T., Burnett M. (2015) A Practical Guide to Controlled Experiments of Software Engineering Tools with Human Participants. *Empirical Software Engineering* 20, 1 (2015), 110–141

[30] Hackman J.R. (1987) The Design of Work Teams. *Handbook of Organizational Behavior,* 315-342.

[31] Eisenhardt, K., (1989). Building theories from case study research, *Academy of management review,* vol. 14, no. 4, pp. 532-550.

[32] Cao, L., Xu, P. (2005) Activity Patterns of Pair Programming, Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005, HICSS '05, DOI: 10.1109/HICSS.2005.66