An Empirical Study on Factors related to Distributed Pair Programming

https://doi.org/10.3991/ijep.v9i2.9947

Despina Tsompanoudi (✉), Maya Satratzemi, Stelios Xinogalos
University of Macedonia, Thessaloniki, Greece
despinats@uom.edu.gr

Leonidas Karamitopoulos
Alexander TEI of Thessaloniki, Thessaloniki, Greece

Abstract—This paper reports students' perceptions and experiences attending an object-oriented programming course in which they developed software using the Distributed Pair Programming (DPP) technique. Pair programming (PP) is typically performed on one computer, involving two programmers working collaboratively on the same code or algorithm. DPP on the other hand is performed remotely allowing programmers to collaborate from separate locations. PP started in the software industry as a powerful way to train programmers and to improve software quality. Research has shown that PP (and DPP) is also a successful approach to teach programming in academic programming courses. The main focus of PP and DPP research was PP's effectiveness with respect to student performance and code quality, the investigation of best team formation strategies and studies of students' attitudes. There are still limited studies concerning relationships between performance, attitudes and other critical factors. We have selected some of the most common factors which can be found in the literature: academic performance, programming experience, student confidence, feelgood factor, partner compatibility and implementation time. The main goal of this study was to investigate correlations between these attributes, while DPP was used as the main programming technique.

Keywords—Pair programming, distributed pair programming

1 Introduction

Distributed Pair Programming (DPP) is a computer programming technique in which programmers develop software remotely using a specialized infrastructure. The aim of this technique is not only to make remote collaboration feasible, but also to gain the advantages of Pair Programming (PP). PP has its origins in the software industry as a part of Extreme Programming and is intended to improve software quality [1]. Typically, it is performed on one computer, involving two programmers working collaboratively on the same code or algorithm. One programmer acts as the "driver" and the other one as the "navigator" (also called "observer"). The driver has possession of keyboard and mouse and types the program code. The navigator reviews the

inserted code and gives guidelines to the driver. PP has been extensively used for enhancing the learning of programming. Research suggests that PP makes learning of programming more pleasant, promotes collaboration and communication between the members of pairs, encourages the sharing of knowledge and skills, and even improves code quality.

Nowadays to participate in a team activity does not require a physical presence anymore. Virtual settings allow real-time communication and collaboration across any distance and at any time [2]. There is also an increasing demand to prepare students for working in interdisciplinary collaborative environments [3]. As an alternative to co-located PP, DPP is performed remotely allowing programmers to collaborate from separate locations [4]. DPP is considered to maintain all the benefits of PP [5] and in addition to allow for the distributed collaboration of pairs from anywhere and at any time. However, an appropriate infrastructure is required to facilitate the process of remote pair programming. Nowadays various types of DPP systems exist in order to cover the different requirements and demands of end-users.

Anecdotal positive feedback from professional programmers inspired researchers to perform educational studies and to incorporate PP in academic courses. As a result, many researchers followed this paradigm and numerous studies were published the following years [6], [7]. The main focus of them was PP's effectiveness with respect to student performance and code quality. Attention was also given to the investigation of best pair formation strategies and studies on students' attitudes [8].

Performance is one of the most investigated factors regarding the effectiveness of PP, indicating that PP has a positive effect on students' grades [8]. Another well-studied factor is pair compatibility. Research suggests that pairing students with similar skill levels has positive results on motivation and participation [9], [10]. Therefore, in order to achieve greater pair compatibility students should be paired according to their programming experience. Similarly, studies showed that pairs' performance is correlated with how comfortable students feel during a PP session (the so-called feel good factor) [11]. Implementation time is also a common measure used in PP studies to evaluate PP's effectiveness [8]. Most of them report that PP requires less time to complete assignments. Finally, students' self-rated confidence has been used to evaluate PP satisfaction, although with contradictory results [5], [12]. Concluding, the most common factors which can be found in the literature are academic performance, programming experience, student confidence, feelgood factor, partner's compatibility and implementation time. The objective of this research was to investigate relationships between these factors that could affect the performance of Computer Science students working on their homework assignments in a DPP environment in the context of an object-oriented programming course.

The remaining article is organized as follows. In the next section follows a presentation of related work in the field. Then, the research questions and the context of the study are presented (Section 3). Section 4 contains the results of the statistical tests. A discussion and conclusions follow in the last section (Section 5).

2 Related Work

Most research studies conducted by academic and industry researchers conclude that PP has positive effects on programmers' performance and software quality. Williams et al. [7] studied several years the application of PP in the classroom. They found that the collaborative nature of PP helps students to achieve advanced learning outcomes, to be more confident and to receive better grades in programming assignments. Other studies indicate that PP leads in higher program quality, continuous knowledge transfer and more student enjoyment [6]. Additionally, PP and DPP are highly acceptable by students when used as the main programming technique in computer science courses [4], [13].

Group formation is considered to be a very important factor that affects the effectiveness of PP, and consequently DPP as well. Pairs can be defined by the instructor or students themselves. In the former case, group formation is based on students' programming skill level, their personality, or even randomly. Relevant studies suggest that pairing students with similar skill levels has positive results on motivation and participation. Toll et al. [14] concluded that the outcomes of PP are better when the skills of the one partner are slightly better or worse than those of the other partner. Williams et al. [9] also concluded that pairs are more compatible if students with similar perceived skill level are grouped together. Katira et al. [15] studied relationships between pair compatibility and personality type, skill level and self-esteem. Among other results, they found that students' perception of their partner's skill level has a significant influence on their compatibility. On the other hand, students' self-esteem does not appear to be a major contributor to pair compatibility.

Lui and Chan [16] investigated pair productivity between pairs of varying skills and experience, such as between novice–novice and expert–expert. They found that novice–novice pairs against novice solos are much more productive than expert–expert pairs against expert solos.

The study of Sfetsos et al. [17] aimed to investigate the impact of pair programmers' personality and temperament on pair performance and pair collaboration. The results showed better performance and collaboration-viability for pairs with heterogeneous personalities and temperaments.

Students' self-rated confidence in programming ability has been used to evaluate PP satisfaction, although with contradictory results. Thomas et al. [12] report that students with less self-confidence seem to enjoy pair programming the most. On the other hand, Hanks [5] states that in his study the most confident students liked pair programming the most, while the least confident students liked it the least. Muller and Padberg [11] define the feel good factor of a pair as how comfortably the developers feel in a pair session. They study correlations between the feelgood factor and pair performance, as well as between programming experience and performance. They found that pair performance is uncorrelated with a pair's programming experience and that the feelgood factor is a candidate driver for the performance of a pair.

Studies in the field of DPP aimed to evaluate effectiveness of DPP in student performance, teaching process, DPP vs. solo programming or DPP vs. co-located PP [18], [19], [20], [21]. Baheti et al. [18] tested productivity and code quality between

DPP and PP and concluded that virtual and co-located teams can produce comparable software. Hanks [20] also showed that students who used a DPP tool performed as well as co-located students on programming assignments and exams. Duque & Bravo [21] compared programmers employing DPP and solo programmers in terms of work productivity and code quality. They found that DPP teams usually spend more time to complete tasks, but make fewer errors than solo programmers. A recent systematic literature review on DPP conducted by da Silva Estácio and Prikladnicki [19] points out those more empirical studies on DPP are needed, especially in industry.

Our previous work in the field lies in the area of DPP. We conducted several evaluations in authentic learning conditions in order to investigate the impact of DPP in a typical Java course. More recently, our research focused on studying factors that might affect student perception and performance while DPP was used as the main programming technique [22]. The study reported here compliments this previous work. In the present work the research questions are answered based on empirical data measured at pair level (pair's performance, pair's implementation time, pair's feelgood factor etc.) while in the previous one at the student level.

As shown, student performance has been extensively studied in the literature using various methods and approaches. In our study we investigate correlations between performance and other critical factors as performed in the previous studies. The results are presented in the following sections.

3 Research Questions and Methodology

3.1 Context of the study

The study presented in this paper was carried out in the context of an undergraduate course on Object-Oriented Programming (OOP) during the academic year 2016-17. The course offers an introduction to the Java programming language and is part of the second-year curriculum. It runs over thirteen weeks with a 3-hour lab class per week. As homework, students were assigned five Java projects to be solved in pairs using a DPP system. Eighty-eight students chose a partner and formed 44 groups.

In order to solve the assignments students had to utilize a DPP system (SCEPPSys) which provides all means for remote collaboration and some logging capabilities [23]. SCEPPSys runs as an Eclipse plugin installed by students and a web-based administration environment used by instructors for monitoring and preparing programming assignments. SCEPPSys includes typical features of DPP systems, such as providing a shared editor, supporting the roles of the driver and navigator, and a text-based communication tool. However, it also includes some unique features that serve specific didactical needs: assignments comprise of small and manageable tasks or steps associated with specific didactical goals or else OOP concepts. SCEPPSys' logging capabilities are related to users' actions and are accessible via the administration environment. Instructors are able to retrieve useful statistical information about submitted assignments and DPP sessions, such as session duration, role switches, exchanged messages and contribution level of each participant. SCEPPSys' logging feature made

it feasible to perform the current and past studies. An overview of the current study is provided in Table 1.

At the end of the course a questionnaire was delivered to the participating students in order to obtain their feedback on the DPP assignments.

Table 1. Course Outline

Course	Object-Oriented Programming
Programming Language	Java
Programming Environment	Eclipse IDE using the SCEPPSys plugin
Academic year	2016-17
Semester	3rd
Duration	13 weeks, 3 hours per week
Participants	88 (44 groups)
Assignments	5 Java projects as homework
Programming approach	Distributed Pair Programming

3.2 Research Questions

The study aimed to answer the following research questions:

RQ1: Is there empirical evidence that pair's performance is dominated by the programming experience of the developers?

RQ2: Is there empirical evidence that pair's performance is dominated by the confidence in programming of the developers?

RQ3: Is there empirical evidence that pairs' performance is dominated by the feelgood factor of the developers?

RQ4: Is there empirical evidence that pair's feelgood factor is dominated by the programming experience of the developers?

RQ5: Is there empirical evidence that pair's feelgood factor is dominated by the confidence in programming of the developers?

RQ6: *Is there empirical evidence that pair's implementation time is dominated by the confidence in programming of the developers?*

RQ7: Is there empirical evidence that pair's performance is dominated by the perceived compatibility of the developers?

The data used in the study was gathered from three different sources: the log files of the DPP system, the questionnaire and students' grades. The variables are summarized in Table 2.

The *performance* is based on the following two measures: the mean grade that pairs had received in the final exam of the OOP course (*PairJavaExam*) and the mean overall grade that they had received for their homework assignments (*PairJavaAssign*).

In order to evaluate *programming experience* student's grades in previous programming courses were considered. More specifically, the average grade from two courses, "Algorithms" and "Procedural Programming", was calculated in order to specify programming experience for each student. For each pair the mean values of prior programming experience was calculated (*PairProgExp*). In addition to that, we

recorded separately the lower value (*LowerProgExp*) and the higher value (*HigherProgExp*) of programming experience for each pair.

Table 2. Factors and variables of the study

Factor	Description	Variable
Performance	Java exam grade Java assignments grade	*PairJavaExam* *PairJavaAssign*
Programming Experience	Grades received in previous programming courses	*PairProgExp* *LowerProgExp* *HigherProgExp*
Confidence	Self-perception in programming interest and ability	*PairConf* *LowerConf* *HigherConf*
Feelgood factor	Degree to which the developers felt comfortable during the pair session	*PairFeelGood* *LowerFeelGood* *HigherFeelGood*
Implementation time	Total time spent in assignments	*PairImpTime*
Compatibility	Pair's compatibility degree based on perceived compatibility	*PairCompatibility* *LowerCompatibility* *HigherCompatibility*

Grades in exams and programming experience were measured on a scale from 0 to 10 (where 10 is "excellent"), while grades in Java assignments were measured on a scale from 0 to 1.5.

Confidence is typically estimated using students' self-assessment in a survey. Just like in the study of Thomas et al. [12] we asked students to place themselves on a scale from 1 ("code-a-phobe") to 9 ("code-warrior"). This scale ranges between students who dislike programming and face difficulties while coding, and students who like and find challenging the programming process. For each pair the mean value of confidence in programming was calculated (*PairConf*). In addition to that, we recorded separately the lower value (*LowerConf*) and the higher value (*HigherConf*) of confidence in programming for each pair.

The *feelgood* factor was determined using students' responses on the question of how they evaluate the overall pair programming experience on a scale

- 1: very bad
- 2: bad
- 3: neutral
- 4: good
- 5: very good

The average value for each pair was then calculated (*PairFeelGood*). In addition to that, we recorded separately the lower value (*LowerFeelGood*) and the higher value (*HigherFeelGood*) of the *feelgood* factor for each pair. This approach is in accordance with the study of Muller and Padberg [11] where the term feelgood factor was first introduced.

The *implementation time* was calculated by the system, and it indicates the total time a pair spent on completing the assignments (*PairImpTime*).

In order to evaluate the impact of pair *compatibility* (*PairCompatibility*) three subgroups were studied:

- Non-compatible pairs
- Compatible pairs
- Very compatible pairs.

The compatibility degree was based on the pair's perceived compatibility. Each student evaluated on a scale from 1 (non-compatible) to 3 (very compatible) how compatible he felt with his partner regarding his programming ability. A pair was defined as non-compatible when both students felt non-compatible with their partner. A pair was defined as high compatible when both students felt very compatible with their partner. All other pairs are considered as compatible pairs. In addition to that, we recorded separately the lower compatibility degree (*LowerComp*) and the higher compatibility degree (*HigherComp*) for each pair.

The type of each one of the above variables is either continuous or ordinal. Data from the log files and students' responses were analyzed in order to run the statistical tests for this study. To measure correlations between the variables we used the Pearson correlation.

The research questions of the current study investigate meaningful relationships between the aforementioned factors. The results are provided in the following section.

4 Results

In this section we present the results of our research questions. First we provide some general results of the studied factors. As presented in Figure 1, the majority of students (79%) evaluated the overall experience in distributed and collaborative solution of assignments as a good (52%) or a very good (27%) experience (feelgood factor). Only 2 students reported a negative experience and both of them were in the same group.

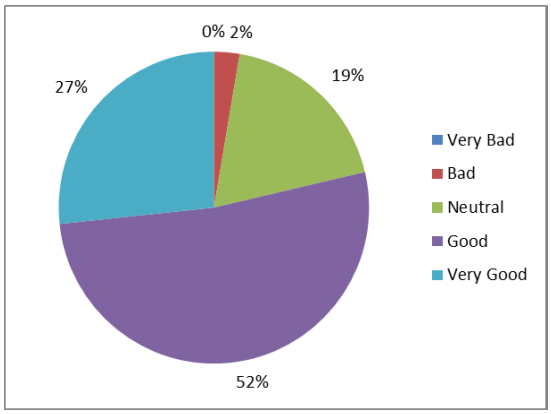

Fig. 1. Overall experience with DPP

Figure 2 depicts the distribution of students' self-confidence in their programming skills. Half of the students rated themselves as "code-warriors" (scale 7-9), 40% of them rated themselves as 4-6 ("middle") and only 10% of the students placed themselves as 1-3 ("code-phobes"). Finally, Figure 3 depicts the distribution of perceived pair compatibility. As shown, students' vast majority report that they were very compatible (49%) or satisfactorily compatible (50%) with their partner.

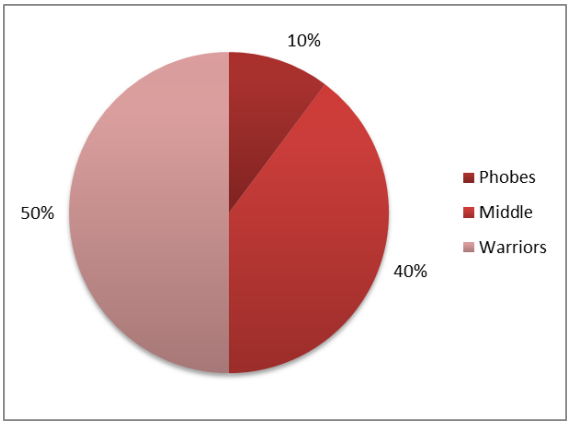

Fig. 2. Distribution of students self-*confidence in Programming*

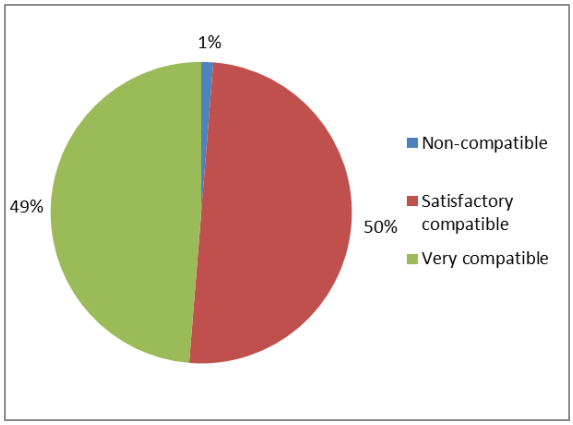

Fig. 3. Perceived Pair Compatibility

RQ1: Is there empirical evidence that pair's performance is dominated by the programming experience of the developers?

The first research question studies the correlation between pair's performance (PairJavaExam, PairJavaAssign) and pair's programming experience (PairProgExp). We found that pair's performance and pair's programming experience are positively correlated (Table 3). The pair with higher programming experience performs better in the Java exam and the Java programming assignments. Although the result might

seem self-evident, we decided to include it in the study as a sort of sanity check to test whether prior programming skills are reflected in the performance of future courses.

Table 3. Results summary of RQ1

RQ1: Results	r	P
PairJavaExam is correlated positively with PairProgExp	0.766	<0.001
PairJavaAssign is correlated positively with PairProgExp	0.511	0.013

Furthermore, we examine whether this correlation is driven by one of the individuals in the pair by focusing on the relationship between:

- The pair's performance (*PairJavaExam*, *PairJavaAssign*) and the programming experience of that member of each pair with the lower experience (*LowerProgExp*);
- The pair's performance (*PairJavaExam*, *PairJavaAssign*) and the programming experience of that member of each pair with the higher experience (*HigherProgExp*).

The results indicate that the Java exam grade of the pair might be driven by both members of the team, whereas, the Java assignments grade of the pair might be driven by the team member who has the higher programming experience (Table 4).

Table 4. Results summary of RQ1 (extended)

RQ1: Results	r	p
PairJavaExam is correlated positively with LowerProgExp	0.722	0.001
PairJavaExam is correlated positively with HigherProgExp	0.781	<0.001
PairJavaAssign is not correlated with LowerProgExp	0.387	0.068
PairJavaAssign is correlated positively with HigherProgExp	0.615	0.002

RQ2*: Is there empirical evidence that pair's performance is dominated by the confidence in programming of the developers?*

This research question studies the relationship between pair's performance (*PairJavaExam*, *PairJavaAssign*) and pair's confidence (*PairConf*). We found that pair's performance and pair's confidence are positively correlated (Table 5). The most confident pair performs better in the Java exam and the Java programming assignments.

Table 5. Results summary of RQ2

RQ2: Results	r	p
PairJavaExam is correlated positively with PairConf	0.715	<0.001
PairJavaAssign is correlated positively with PairConf	0.355	0.031

Furthermore, we examine whether this correlation is driven by one of the individuals in the pair by focusing on the relationship between:

- The pair's performance (*PairJavaExam*, *PairJavaAssign*) and the confidence level of that member of each pair with the lower confidence (*LowerConf*);
- The pair's performance (*PairJavaExam*, *PairJavaAssign*) and the confidence level of that member of each pair with the higher confidence (*HigherConf*).

The results indicate that the Java exam grade of the pair might be driven by both members of the team, whereas, the Java assignments grade of the pair might be driven by the team member who has the higher confidence (Table 6):

Table 6. Results summary of RQ2 (extended)

RQ2: Results	r	p
PairJavaExam is correlated positively with LowerConf	0.661	<0.001
PairJavaExam is correlated positively HigherConf	0.673	<0.001
PairJavaAssign is not correlated with LowerConf	0.274	0.101
PairJavaAssign is correlated positively with HigherConf	0.387	0.018

RQ3: Is there empirical evidence that pairs' performance is dominated by the feelgood factor of the developers?

This research question studies the relationship between pair's performance (PairJavaExam, PairJavaAssign) and pair's feelgood factor (PairFeelGood). In our study, pair's performance does not correlate with the feelgood factor of the developers (PairFeelGood) (Table 7).

Table 7. Results summary of RQ3

RQ3: Results	r	p
PairJavaExam is not correlated with PairFeelGood	-0.037	0.859
PairJavaAssign is not correlated with PairFeelGood	0.026	0.885

It is natural to ask whether the same holds for the feelgood factor of the individual developers in a pair. We analyzed the relationship between:

- The pair's performance (*PairJavaExam*, *PairJavaAssign*) and the feelgood factor of that member of each pair who felt less comfortable with the pair programming situation than the other member (*LowerFeelGood*);
- The pair's performance (*PairJavaExam*, *PairJavaAssign*) and the feelgood factor of that member of each pair who felt more comfortable with the pair programming situation than the other member (*HigherFeelGood*).

The results indicate that there is no correlation between these individual feelgood factor levels and pair's performance (Table 8).

Table 8. Results summary of RQ3 (extended)

RQ3: Results	r	p
PairJavaExam is not correlated with LowerFeelGood	-0.022	0.918
PairJavaExam is not correlated with HigherFeelGood	-0.046	0.827
PairJavaAssign is not correlated with LowerFeelGood	0.047	0.791
PairJavaAssign is not correlated with HigherFeelGood	0.000	1.000

RQ4: Is there empirical evidence that pair's feelgood factor is dominated by the programming experience of the developers?

This research question studies the relationship between pair's feelgood factor (PairFeelGood) and pair's programming experience (PairProgExp). In our study, no correlation was found between pair's feelgood factor and pair's programming experience (r=0.034, p=0.897).

It is natural to ask whether the same holds for the programming experience of the individual developers in a pair. We analyzed the relationship between:

- The pair's feelgood factor (PairFeelGood) and the programming experience of that member of each pair with the lower experience (LowerProgExp);
- The pair's feelgood factor (PairFeelGood) and the programming experience of that member of each pair with the higher experience (HigherProgExp).

The results indicate that there is no correlation between these individual levels of programming experience and pair's feelgood factor (Table 9).

Table 9. Results summary of RQ4 (extended)

RQ4: Results	r	p
PairFeelGood is not correlated with LowerProgExp	0.163	0.533
PairFeelGood is not correlated with HigherProgExp	-0.191	0.462

RQ5: Is there empirical evidence that pair's feelgood factor is dominated by the confidence in programming of the developers?

This research question studies the relationship between pair's feelgood factor (PairFeelGood) and pair's confidence in programming (PairConf). In our study, no correlation was found between pair's feelgood factor and pair's confidence in programming (r=-0.196, p=0.268).

It is natural to ask whether the same holds for the programming experience of the individual developers in a pair. We analyzed the relationship between:

- The pair's feelgood factor (PairFeelGood) and the confidence in programming of that member of each pair with the lower confidence (LowerConf);
- The pair's feelgood factor (PairFeelGood) and the confidence in programming of that member of each pair with the higher confidence (HigherConf).

The results indicate that there is no correlation between these individual levels of programming confidence and pair's feelgood factor (Table 10).

Table 10. Results summary of RQ5 (extended)

RQ5: Results	R	p
PairFeelGood is not correlated with LowerConf	-0.226	0.199
PairFeelGood is not correlated with HigherConf	-0.080	0.652

RQ6: Is there empirical evidence that pair's implementation time is dominated by the confidence in programming of the developers?

This research question studies the relationship between pair's implementation time (PairImpTime) and confidence in programming (PairConf).We found that pair's implementation time and pair's confidence in programming (PairConf) are correlated negatively (r=-0.359, p=0.029). This means that pairs with a high level of confidence on their programming skills needed less time to complete the assignments.

It is natural to ask whether the same holds for the confidence in programming of the individual developers in a pair. We analyzed the relationship between:

- The pair's implementation time (PairImpTime) and the confidence in programming of that member of each pair with the lower confidence (LowerConf);
- The pair's implementation time (PairImpTime) and the confidence in programming of that member of each pair with the higher confidence (HigherConf).

The results indicate that the implementation time of the pair might be driven by the team member who has the higher self-confidence in programming (Table 11).

Table 11. Results summary of RQ6 (extended)

RQ6: Results	r	p
PairImpTime is not correlated with LowerConf	-0.186	0.271
PairImpTime is correlated with HigherConf	-0.549	<0.001

RQ7: Is there empirical evidence that pair's performance is dominated by the perceived compatibility of the developers?

This research question studies the relationship between pair's performance (PairJavaExam, PairJavaAssign) and pair's compatibility (PairCompatibility). We found that pair's performance in Java exam (PairJavaExam) and pair's compatibility (PairCompatibility) are correlated positively whereas pair's performance in Java assignments (PairJavaAssign) and pair's compatibility (PairCompatibility) are not correlated (Table 12). This means that pairs with higher perceived compatibility performed better in Java exam.

Table 12. Results summary of RQ7

RQ7: Results	r	p
PairJavaExam is correlated positively with PairCompatibility	0.472	0.011
PairJavaAssign is not correlated with PairCompatibility	0.251	0.133

It is natural to ask whether the same holds for the perceived compatibility of the individual developers in a pair. We analyzed the relationship between:

- The pair's performance (*PairJavaExam*, *PairJavaAssign*) and the perceived compatibility of that member of each pair with the lower compatibility (*LowerCompatibility*);
- The pair's performance (*PairJavaExam*, *PairJavaAssign*) and the perceived compatibility of that member of each pair with the higher compatibility (*HigherCompatibility*).

The results indicate that the Java exam grade of the pair might be driven by that member of each pair with the lower compatibility, whereas, the Java assignments grade is not driven by either one of the members in team (Table 13).

Table 13. Results summary of RQ7 (extended)

RQ7: results	r	p
PairJavaExam is correlated positively with LowerCompatibility	0.433	0.021
PairJavaExam is not correlated with HigherCompatibility	0.368	0.054
PairJavaAssign is not correlated with LowerCompatibility	0.261	0.119
PairJavaAssign is not correlated with HigherCompatibility	0.164	0.334

5 Discussion and Conclusion

The benefits of PP are numerous and have been extensively recorded in the literature. Studies on PP are more exhaustive than those on DPP, and many of the factors investigated about PP have not yet been examined for DPP. In this paper we investigate correlations between performance, programming experience, student confidence, feelgood factor, partner compatibility and implementation time while DPP was used as the main programming technique. Based on our findings we can draw some conclusions and provide a practical contribution to the literature.

In this empirical study teamwork in the form of DPP-assignments was once again evaluated very positively. The majority of students (79%) evaluated the overall experience in distributed and collaborative solution of assignments as a good or a very good experience. In previous studies DPP has always gained positive feedback from students. A similar evaluation result is also presented in the work of Muller and Padberg [11].

According to the results pair's prior programming experience is associated with pair's performance in an OOP course that is supported by DPP assignments. There is a statistically significant correlation between previous programming performance and overall performance in Java (exam and assignments grade). Since prior knowledge of each pair member is a determinant factor for their learning efficiency, students should have a deep knowledge on fundamental programming concepts or constructs before enrolling in an OOP course.

The same holds for pair's performance and pair's confidence in programming. The most confident pair performs better in Java exam and Java programming assignments.

Concerning pair' implementation time and pair's programming confidence these two factors are associated.

We also found that pair's Java assignments grade might be driven by the team member with the higher confidence and the higher programming experience. The same holds for pair's implementation time. Pair's implementation time might be driven by the team member who has the higher self-confidence in programming.

The results concerning feelgood factor show that pair's performance isn't dominated by the feelgood factor. The results also suggest that how comfortably team members felt during DPP sessions is not dominated by pair's prior programming experience or pair's programming confidence. Thus, students seem to feel comfortable during DPP sessions regardless of their perception in programming competence or their prior programming experience.

Finally we found that only pair's performance in Java exam is correlated with pair's compatibility.

Taking into consideration all the above presented results we could conclude that the most dominant factors for the pair's performance is prior programming experience and confidence in programming and not how comfortably students feel during the DPP sessions. We also have some indication that pair's Java assignments grade is driven by the team member with the higher programming experience and higher confidence whereas pair's Java exam grade is driven by the team member with the lower perceived compatibility.

Finally we should point out that the fact that a correlation exists is not sufficient to conclude that the prior programming experience or the perception on programming competence actually drives the pair performance. A valid conclusion is, though, that the pair prior programming experience or the perception in programming competence are candidate drivers for the performance of a pair. For the same reason a valid conclusion is that pair's compatibility is a candidate driver for the performance of a pair on the final exam.

6 Acknowledgement

This research is funded by the University of Macedonia Research Committee as part of the "Principal Research 2019" funding program.

7 References

[1] Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. IEEE software, 17(4), 19-25. https://doi.org/10.1109/52.854064

[2] Dávideková, M., & Hvorecký, J. (2017). ICT Collaboration Tools for Virtual Teams in Terms of the SECI Model. In International Journal of Engineering Pedagogy (IJEP), Vol. 7, No. 1, 2017 https://doi.org/10.3991/ijep.v7i1.6502

[3] Rodriguez, J., & Esparragoza, I. E. (2017). Motivation of Engineering Students Participating in Multinational Design Projects – Comparison Based on Gender and Class Status. In International Journal of Engineering Pedagogy (IJEP), Vol. 7, No. 4, 2017. https://doi.org/10.3991/ijep.v7i4.7516

[4] Zacharis, N. (2009). Evaluating the effects of virtual pair programming on students' achievement and satisfaction. International Journal Of Emerging Technologies In Learning (IJET), 4(3), 34-39. https://doi.org/10.3991/ijet.v4i3.772
[5] Hanks, B. (2006). Student attitudes toward pair programming. In ACM SIGCSE Bulletin (Vol. 38, No. 3, pp. 113-117). ACM. https://doi.org/10.1145/1140123.1140156
[6] Faja, S. (2011). Pair Programming as a Team Based Learning Activity: A Review of Research. Issues in Information Systems, XII(2), 207–216.
[7] Williams, L., McCrickard, D. S., Layman, L., Hussein, K. (2008). Eleven Guidelines for Implementing Pair Programming in the Classroom. In Proceedings of the Agile 2008 (AGILE '08), 445–452. https://doi.org/10.1109/Agile.2008.12
[8] Salleh, N., Mendes, E., & Grundy, J. (2011). Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. IEEE Transactions on Software Engineering, 37(4), 509-525. https://doi.org/10.1109/TSE.2010.59
[9] Williams, L., Layman, L., Osborne, J., & Katira, N. (2006). Examining the compatibility of student pair programmers. In Agile Conference, 2006. IEEE. https://doi.org/10.1109/AGILE.2006.25
[10] Braught, G., MacCormick, J., & Wahls, T. (2010). The benefits of pairing by ability. In Proceedings of the 41st ACM technical symposium on Computer science education (pp. 249-253). https://doi.org/10.1145/1734263.1734348
[11] Muller, M. M., & Padberg, F. (2004). An empirical study about the feelgood factor in pair programming. In Software Metrics, 2004. Proceedings. 10th International Symposium on (pp. 151-158). IEEE. https://doi.org/10.1109/METRIC.2004.1357899
[12] Thomas, L., Ratcliffe, M., & Robertson, A. (2003). Code warriors and code-a-phobes: a study in attitude and pair programming. In ACM SIGCSE Bulletin (Vol. 35, No. 1, pp. 363-367). ACM. https://doi.org/10.1145/792548.612007
[13] Nan, I., Kau, B., & Rugelj, J. (2008). Pair programming as a modern method of teaching computer science. International Journal of Emerging Technologies in Learning (IJET), 3, 45-49.
[14] Van Toll, T., Lee, R., & Ahlswede, T. (2007). Evaluating the usefulness of pair programming in a classroom setting. In Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on (pp. 302-308). IEEE.
[15] Katira, N., Williams, L., Wiebe, E., Miller, C., Balik, S., & Gehringer, E. (2004). On understanding compatibility of student pair programmers. In ACM SIGCSE Bulletin (Vol. 36, No. 1, pp. 7-11). ACM. https://doi.org/10.1145/1028174.971307
[16] Lui, K. M., & Chan, K. C. (2006). Pair programming productivity: Novice–novice vs. expert–expert. International Journal of Human-computer studies, 64(9), 915-925. https://doi.org/10.1016/j.ijhcs.2006.04.010
[17] Sfetsos, P., Stamelos, I., Angelis, L., & Deligiannis, I. (2009). An experimental investigation of personality types impact on pair effectiveness in pair programming. Empirical Software Engineering, 14(2), 187. https://doi.org/10.1007/s10664-008-9093-5
[18] Baheti, P., Gehringer, E., & Stotts, D. (2002). Exploring the efficacy of distributed pair programming. In Conference on Extreme Programming and Agile Methods (pp. 208-220). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45672-4_20
[19] da Silva Estácio, B. J., & Prikladnicki, R. (2015). Distributed pair programming: A systematic literature review. Information and Software Technology, 63, 1-10. https://doi.org/10.1016/j.infsof.2015.02.011
[20] Hanks, B. (2008). Empirical evaluation of distributed pair programming. International Journal of Human-Computer Studies, 66(7), 530-544. https://doi.org/10.1016/j.ijhcs.2007.10.003

[21] Duque, R., & Bravo, C. (2008). Analyzing work productivity and program quality in collaborative programming. In Software Engineering Advances, 2008. ICSEA'08. The Third International Conference on (pp. 270-276). IEEE. https://doi.org/10.1109/ICSEA.2008.82
[22] Satratzemi, M., Xinogalos, S., Tsompanoudi, D., & Karamitopoulos, L. (2018). Examining Student Performance and Attitudes on Distributed Pair Programming. Scientific Programming, 2018. https://doi.org/10.1155/2018/6523538
[23] Tsompanoudi, D., Satratzemi, M., & Xinogalos, S. (2015). Distributed Pair Programming using Collaboration Scripts: An Educational System and initial Results. Informatics in Education, Vol. 14, No. 2, 291–314, 2015. https://doi.org/10.15388/infedu.2015.17

8 Authors

Despina Tsompanoudi is member of the Software and Data Engineering Lab, Department of Applied Informatics, University of Macedonia, Egnatia 156, 54006 Thessaloniki, Greece. She received a BSc in Informatics from the Aristotle University of Thessaloniki in 2003 and an MSc in Information Systems from the University of Macedonia in 2006. She holds a PhD degree in Computer Science from University of Macedonia, Greece, under the supervision of Professor Maya Satratzemi. She is a Computer Science teacher since 2005 and currently teaches in Eleftheroupoli High School, Greece.

Maya Satratzemi is a professor at the Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece. She was awarded the BS degree in Mathematics from the Aristotle University of Thessaloniki in 1980 and the PhD degree in Informatics in 1991 from the department of Applied Informatics, University of Macedonia. Her current main research interests lie in the area of Educational Programming Environments and Techniques, Adaptive and Intelligent Systems, Collaborative Learning Systems, and Game-based Learning. She has published a significant number of papers in international journals and in International and National conferences. She was Conference co-chair of the 8th ITiCSE (ACM). maya@uom.edu.gr

Stelios Xinogalos is an assistant professor at the Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece. He was awarded the BS degree in Applied Informatics from the University of Macedonia in 1998 and the PhD degree in Informatics in 2002 from the department of Applied Informatics, University of Macedonia. His current main research interests lie in the area of Programming Environments and Techniques, Object-Oriented Design and Programming, Educational Programming Environments, and Serious Games. He has published a significant number of papers in International journals and in International and National conferences.stelios@uom.edu.gr

Leonidas Karamitopoulos is a lecturer at the Department of Information Technology, Alexander Technological and Educational Institute of Thessaloniki, Greece. He was awarded the B.A. degree in Mathematics from the Aristotle University of Thessaloniki in 1990, the M.Sc. degree in Operations Research from the George Mason University, U.S.A., and the Ph.D. degree in Data Mining and Knowledge Discovery in Time Series in 2009 from the department of Applied Informatics, University of Macedonia. His current main research interests lie in the area of data mining in time

series and recommendation systems. He has published a number of papers in international journals and in International and National conferences. He was member of organization committee of the 10th Balkan Conference of Operations Research. lkaramit@otenet.gr

This article is a revised version of a paper presented at the International Conference on Interactive Collaborative Learning (ICL2018), held September 2018, in Kos, Greece. Article submitted 2018-11-30. Resubmitted 2019-01-30. Final acceptance 2019-01-30. Final version published as submitted by the authors.