Factors affecting students' performance in Distributed Pair Programming

Stelios Xinogalos, Maya Satratzemi, Alexander Chatzigeorgiou, Despina Tsompanoudi

Department of Applied Informatics, School of Information Sciences, University of Macedonia

Abstract

Pair Programming has been shown to increase productivity and code quality not only in professional software development but also in the context of programming education. The provision of broadband internet access gave rise to Distributed Pair Programming (DPP) enabling two programmers to collaborate remotely. To gain insight into the benefits of DPP we performed an empirical study on an object-oriented programming (OOP) course where 62 students carried out assignments through a DPP platform. The goal of the study is to investigate, in the context of DPP, whether prior programming skills (assessed at the level of student, his/her partner and pair) and pair compatibility are related to student performance. To further examine the effect of DPP on learning outcomes we have studied whether a pair's performance on DPP assignments is related to the students' grade. The findings indicate that the student's actual skill and the pair's actual skill affect his/her performance in an OOP course. The results also suggested that there is no association between pair compatibility and his/her own performance. Finally, pair performance on DPP assignments is related to the individual student performance in the final exams. Such evidence can be used to guide instructors when planning DPP assignments and especially when forming student pairs.

Keywords: Distributed pair programming, collaborative learning tools, object-oriented programming, computer science education

1. INTRODUCTION

Pair Programming (PP) has attracted increasing interest among researchers in diverse educational institutions around the world. A significant number of them (Williams, 2007; Hanks et al., 2011) have conducted numerous studies concerning the effects of PP. Early research on the use of PP as a pedagogical tool focused mainly on whether it was capable of benefitting students as regards their productivity and the quality of the work produced (Williams et al., 2000; McDowell et al., 2003; Mendes et al., 2006, Naggappan et al., 2003). The findings demonstrated 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 research has shown that PP has a positive effect on higher program quality, continuous knowledge transfer, higher student enjoyment (Faja, 2011), generally better attitudes toward programming and computer science, as well as a higher capacity to retain knowledge in follow on computer science courses (Braught et al., 2011). A Systematic Literature Review (SLR) of empirical studies was carried out by Salleh et al. (2011) in order to investigate which factors influence the effectiveness of computer science (CS) and software engineering (SE) studies by measuring the performance of PP on CS/SE students. Their research aimed to improve the practice of PP as a pedagogical tool in CS/SE education by investigating pair compatibility and its effect on PP's effectiveness. Pair compatibility is a factor that significantly influences the progress and outcome of collaboration, with many studies suggesting pairing students based on similar skill levels (Faja, 2011). The majority of students reported that one major shortcoming of PP was scheduling conflicts (Faja, 2011).

In recent years a number of studies have focused on Distributed Pair Programming (DPP) and its effects on learning outcomes. While retaining most of the benefits of PP, DPP has a beneficial effect on remote collaboration in distance education (Hanks, 2008; Stotts et al., 2003). da Silva Estácio and Prikladnicki (2015) conducted an SLR on DPP in order to identify the necessary settings for the adoption of DPP, as well as the proposed tools in the literature. While the papers included in the SLR investigated DPP from the perspectives of teaching and practice, the majority of the 34 primary studies dealt with it from the teaching domain. The main variables found in the literature regarding DPP with the focus on teaching were: performance, grades, productivity, motivation, confidence, and learning. The results of their study showed that a) DPP requires a specific infrastructure, but the existing studies do not explore the impact of the distribution in the details; and b) many tools are proposed to support DPP practice, but few of them have been evaluated within a software development team.

In order to apply DPP in our department and study its effects on students' learning outcomes and the factors that could affect DPP, we developed an Eclipse plugin named SCEPPSys (Tsompanoudi et al., 2015). Since 2013-2014 we have used SCEPPSys in our department and we have conducted 3 studies (Tsompanoudi et al., 2013; Tsompanoudi et al., 2015; Tsompanoudi et al. 2016) in order to examine the different aspects of DPP on CS students. In the last two experiments, our research focused on the study of various ways to switch the roles of the driver that is writing code and the navigator that reviews and comments on the code (turn-taking policies) and their effects on student performance. More information about these policies and the features of SCEPPSys is provided in Section 4.

The literature results showed that 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 present the findings of a new study conducted on a third semester CS course. The students worked remotely in pairs on their homework assignments of an object-oriented programming course in Java using SCEPPSys. This work focused on examining possible factors which we felt had the potential to have an effect on student performance when working on DPP. The goal of the study is to investigate, in the context of DPP, whether prior programming skills assessed at the level of student, his/her partner, of the pair as a whole and pair compatibility are related to student performance. To further examine the effect of DPP on learning outcomes we have studied whether pair's performance on DPP assignments is related to students' grade. Our study contributes on the role of pair compatibility -in terms of students' prior programming skill- on student performance and consequently on effective pair formation. Results are drawn based on the data collected by our system SCEPPSys (number of projects submitted, code contribution rate of each partner), assignment and exam grades, as opposed to other studies which are based on questionnaires completed by students.

The remainder of this article is organized as follows: Section 2 gives the background and related studies. In Section 3 we define the Research Questions of our study. In Section 4 we describe the case study, while Section 5 provides the results of the study and Section 6 discusses the results. Finally, Section 7 presents the threats to the validity of this study and Section 8 draws the conclusions.

2. RELATED WORK

The model of PP originated from the software industry as a part of Extreme Programming (XP). It involves two programmers working at the same workstation and sharing one computer in order to develop software. When applied as a part of XP, pair programming has shown to produce faster program code with fewer bugs and better quality (Williams et al., 2000). In order to support novice programmers PP has been successfully incorporated in academic courses. As a form of collaborative learning, PP involves close collaboration, knowledge transfer, negotiation and sharing of programming skills. Novice programmers not only receive the advantages of group work, they also experience agile software development techniques used on the market. Research findings suggest that collaborative problem solving helps students to be more confident, to enjoy programming and to improve academic performance (Williams et al., 2000). However, group work has its limitations and many studies have investigated factors inhibiting PP. The most common problems are non-participation, unequal workload distribution and pair incompatibility (Williams, 2007; Schümmer and Lukosch, 2009). Consequently a considerable number of studies investigated various team formation strategies (Katira et al., 2004; Sfetsos et al., 2009; Mendes et al., 2006). Salleh et al. (2011) performed a SLR of empirical studies that investigated factors influencing the effectiveness of PP and studies that measured the effectiveness of PP for CS/SE students. One of the research questions was to investigate the compatibility factors that affect PP's effectiveness as a CS/SE pedagogical tool, and which pairing configurations were considered as most effective. Compatibility factors are factors believed to influence the affinity of students when working in pairs. Fourteen (14) factors were identified in 17 studies which investigated how these affected or correlated with PP's effectiveness and/or pair compatibility. The 14 factors were: personality, actual skill level, perceived skill level, self-esteem, gender, ethnicity, learning style, work ethic, time management ability, "feelgood" factor, confidence level, type of role, type of tasks, and communication skills. PP effectiveness was measured using various factors, organized in four categories: technical productivity, program/design quality, academic performance, and satisfaction. Technical productivity was the most common method used to assess the effectiveness of PP, followed by program/design quality. A subset of 16 studies evaluated PP effectiveness based on students' academic performance in final exams, midterms, assignments, and course grades. Besides the objective measurements, PP effectiveness was evaluated subjectively in 22 studies using students' perceived satisfaction in their experience of PP sessions.

The two compatibility factors investigated most were skill level and personality type. Five PP studies regarded actual paired skill level as one of the determinant factors of PP effectiveness (Canfora et al., 2005; Cliburn, 2003; Katira et al., 2004; Van Toll et al., 2007; Williams et al., 2006). The actual skill level was determined based on programming experience, academic background, and students' academic performance.

Canfora et al. (2005) executed two experiments with students of different academic background. The goal of the experiments was to evaluate how individual background affects the knowledge built when designing in pairs. All the pairs were formed randomly. Their conclusions were that forming pairs from the same educational background emphasized the expected benefits of pair design. Coupling a person with a scientific background and one with a non-scientific background does not improve the performance of the latter and actually worsens the performance of the former.

Cliburn (2003) discussed his experiences with pair programming in an introductory programming course and he presented the revised process used to make it more successful. He paired students with similar course grades. Students worked in pairs for labs and homework assignments. The conclusions drawn were that assigning partners so that students always work with someone at or near the same skill level seems to be the best strategy for ensuring that students do not get through the course only on the strengths of their collaborators.

Katira et al. (2004) performed their research on three different courses: Introduction to Programming – Java (CS1), undergraduate Software Engineering (SE), and graduate object-oriented (OO) programming. The CS1 and SE students worked in pairs, with different assigned partners for each one of their assignments. Students of the OO course could choose to work in pairs or to work alone; a teaching assistant paired students. After each pairing cycle, all students completed a web-based peer-evaluation survey about the contributions of their collaborator for the assignment and how compatibly the pair worked together. The researchers in their study tested hypotheses concerning factors on compatibility. The results of the Spearman's rank-order correlation suggested that there is no association between the actual skill level and the compatibility of the CS1 and the SE students. However, there was a strong positive correlation between the actual skill level and the compatibility of the graduate OO students.

Van Toll et al. (2007) investigated which difference in skill level between programmers leads to the best learning experience for each programmer. A student participated in pair programming in 4 different types of PP assignments, i.e. Programming with less experience, Programming with more experience, Programming with significantly more experience, Programming with equal experience. They found that pair programming works best when the pairs are of slightly different skill level. The best learning experience for each programmer was created when one partner had slightly more, or slightly less skill than the other. However, when the difference in skill is too great, the pair programming ideal can break down.

Williams et al. (2006) conducted experiments focused on examining possible factors that they felt had the potential to affect compatibility between student pair programmers. All students were required to complete a web-based peer evaluation survey about the contributions and compatibility of their collaborator after each paired assignment was completed. The hypothesis was: whether pairs were more compatible when students with similar actual skill level were grouped together. The absolute differences in the collaborators' midterm grades were examined for a correlation with compatibility. Their results showed that the most effective pairing configuration is to pair students of a similar competency level using as a basis their exam scores/GRE/GPA or programming experience. The researchers recommended that educators who are willing to practice PP in their classroom should pair students according to their skill or competency level to achieve greater pair compatibility.

Plonka et al. (2012) reported an empirical study that investigates disengagement. They qualitatively analyzed video and audio recording of industrial PP sessions in order to investigate circumstances that lead to disengagement. They identified five reasons for disengagement: interruptions during the collaboration, the way the work is divided, the simplicity of the task involved, social pressure on inexperienced pair programmers, and time pressure.

An early study of DPP was conducted by Baheti, et al. (2002). They studied a graduate-level object-oriented programming course in which students worked on team projects. They didn't find any differences between the collocated and distributed teams in terms of productivity or software quality.

Canfora et al. (2003) held two experiments aimed to study the impact of the DPP according to productivity metrics time and code quality (students' grades). The results indicated that the pairs tended to work alone. They identified four factors in order to explain the results: failure to establish a working protocol, the conflict of ideas among the pairs, problems with the chat software and different levels of experience between the pairs. After their second experiment they reported two factors to the DPP success: appropriate communication and collaboration support.

Hanks (2008) developed and empirically evaluated a tool specifically to support DPP. In his study, students were randomly paired into two groups. Students who used DPP performed equivalently, in terms of grades on programming assignments and the final exam with students who used PP.

Duque (2008) evaluating COLLECE a tool for DPP reported that code quality of students who enjoyed DPP was higher than the solo programmers, measuring by the syntactical errors. On the other hand, DPP required more time to complete a task, spending more effort in communication and coordination interactions between the students.

Schümmer and Lukosch (2009) evaluated XPairtise over an 18-week period, with the focus on studying student interactions, emphasizing users' contributions and communications. Besides general observations of DPP they tested whether a) students with comparable skills had frequent role switches b) in a pair with different experience level the experienced partner kept the driver role throughout the session and c) in both above cases driver and navigator frequently used the feature of XPairtise which supports collaboration awareness. The first hypothesis was not confirmed while the second was confirmed and the third was confirmed only for the driver.

Rosen et al. (2010) described the establishment of DPP in a corporate project kick-off. In their case study they reported benefits such as a high level of communication, knowledge transfer and productive work as well as reduced delay in feedback supported the distributed collaboration and hence the project kick-off. Some of the negative effects were conflicts in role fulfillment (due to little experience with PP/DPP, little role consciousness, diverging expectations), ambiguity in session goals (due to lack of precise communication, diverging project goals, short notice of session planning) and missing awareness (due to weaknesses in the technical infrastructure, non-use of awareness functionalities, lack of talk-aloud).

Zacharis (2011) conducted a study investigating the effectiveness of virtual pair programming (VPP) on student performance and satisfaction in an introductory Java course. Two groups of students were formed, VPP students and solo students. In this study two factors were examined: code productivity and software quality. Besides that he compared the midterm and final examination scores of VPP and solo students.

Finally, a survey of students' perceptions of VPP was administered. Positive results were reported for the effectiveness of VPP.

da Silva Estácio and Prikladnicki (2015) conducted a research on DPP, the motivation of which was the lack of studies about the use of DPP in industry. In their SLR on DPP, they reported that most of the studies they found concerned PP and only 22 papers were related to DPP. In addition, there were few papers describing case studies about the adoption of DPP in industry. Most of the studies about DPP concentrated on tool proposals. They gathered data from a field study concerning variables (code quality, team productivity and communication, difference of knowledge between the pairs), DPP aspects (company guidelines for using DPP, infrastructure and methods for DPP, development tool for DPP, facilitator to support DPP, experience between pairs of DPP), benefits (execution time, motivation), challenges of using DPP (collaboration and communication challenges), and opinion (suggestions). Based on the literature review and the field study they concluded by suggesting twelve practices that could help professionals in the practice of DPP.

Schenk et al. (2014) conducted one of the few qualitative researches on the literature of DPP, based on the direct observation of a single, highly competent distributed pair of industrial software developers. They used Saros an open source Eclipse plugin for DPP which lets the collaborate team members concurrently view or edit the same or different lines in the Shared Editor. This feature offers the freedom of concurrent editing and has even more awareness deficits than screen sharing. The aim of their research concerned to study the effects of the awareness issues and the use and effects of editing freedom. The reported results were that skilled pairs may bridge the awareness deficits without visible obstruction of the overall process and also may use the additional editing freedom in a useful limited fashion, resulting in potentially better fluency of the process than local pair programming.

Summarizing the research on DPP, previous studies have focused on factors concerning code productivity, software quality, time, communication or evaluation of advanced features concerning features of Integrated Development Environments (IDEs). Research on PP is more mature and different types of factors have been investigated in order to study the effectiveness of PP. However, the literature lacks studies on the effectiveness of DPP as a CS/SE pedagogical tool. Moreover, there is no research evidence on which pairing configurations in the context of DPP are the most effective in terms of achieved academic performance.

Our current work was drawn from the studies on PP effectiveness and we especially examined whether actual skill at the level of student, of his/her partner, of the pair as a whole and pair compatibility are related to student performance. As can be seen from the above presented literature review on DPP, a similar study has not been conducted. In addition, studies on PP examined actual skill in relation to pair compatibility. These studies have examined pair compatibility using a subjective way whereas our study defined pair compatibility in an objective way based on the prior programming performance of a pair.

The literature review on DPP has shown that some factors which have been examined in the PP context, such as code quality, productivity etc. were also investigated in the DPP context. The difference between DPP and PP is DPP's allowing geographically-distributed teams to collaborate and share program code, and thus such collaboration is only feasible if an underlying infrastructure enables for all necessary interactions. Although IDE's for DPP cover the basic requirements of DPP, most of them cannot address common problems of PP, such as unequal contributions from each member of a pair, feedback during DPP sessions, communication problems. Even though DPP is an alternative to PP, these two techniques are not the same, and thus factors which have been examined in the PP context could also be examined in the DPP context, since DPP is more demanding because the members of the pair are not co-located and their common work is dependent on the features of the IDE and an infrastructure that has to be set up and configured by students themselves. Pair compatibility in particular, might have a different effect on distributed programming, because for pairs with differences in their programming skills disengagement of the less skilled member might occur more frequently than in a PP context.

3. RESEARCH OBJECTIVE

The objective of this research is to investigate the factors that could affect the performance of CS students working on their homework assignments in a DPP environment in the context of an OOP course. Among 17 studies which have investigated how various factors affected or are correlated with PP's effectiveness and/or pair compatibility, 10 studies considered actual skill level, 9 studies concerned personality type and only 3 studies examined additional factors. Given that in the context of PP the most commonly investigated factor concerning effectiveness is actual skill, we chose to examine if this predominant factor for PP

effectiveness also has a bearing on DPP effectiveness. Thus relying mainly on the results of previous studies on PP, our focus was to examine if the student actual skill, pair actual skill and pair compatibility are determinant factors of DPP effectiveness. The actual skill of each student is determined based on student's prior programming performance i.e. his/her prior knowledge in C language. We must note that the grade in a prior programming course is not the only parameter reflecting the students' actual skill level, but actual skill level has been assessed by grades in similar studies as well (Katira et al., 2004; Williams et al., 2006; Van Toll et al., 2007). Pair actual skill and pair compatibility is determined based on the pair's prior programming skills. Williams et al. (2006) in their study examined whether "*Pairs are more compatible if students with similar actual skill level are grouped together*". As an objective measure for pair actual skill, the researchers used the absolute differences in the partners' midterm grades, SAT, GRE, and overall GPA for a correlation with compatibility. In this study compatibility was defined in a subjective manner. The students were asked to evaluate their perception of their partner's skill level and their joint compatibility on an ordinal scale. In our study, we measured pair actual skill in a similar way as Williams et al. (2006) did; however, compatibility was measured using an objective manner that is described in subsection 4.3. Academic performance on an OOP course was used to measure DPP effectiveness. The metrics used to measure academic performance are exam grade, assignments grade, the average code contribution rate in the programs developed during the homework assignments and the number of assignments submitted by pairs. The presented empirical study falls under the category of case studies, as our goal was to establish relationships between monitored attributes by collecting and analyzing data, in a particular setting, for a particular period of time (Wohlin et al., 2000).

This study will therefore address the following research questions:

RQ1: Does the student's actual skill affect his/her performance on an OOP course that is supported by DPP assignments?

Although RQ1 might seem self-evident, we decided to include it in the study as a sort of sanity check to test whether the sample included in the study met this fundamental assumption that prior skills are reflected in the performance of future courses.

RQ2: Does the partner's actual skill affect student performance on an OOP course that is supported by DPP assignments?

RQ3: Does the pair's actual skill and the pair's compatibility affect student performance on an OOP course that is supported by DPP assignments?

To further investigate if carrying out assignments in a DPP environment has any relation to overall student performance on an OOP course, the following research question was formulated:

RQ4: Does the pair's performance on DPP assignments affect student performance on an OOP course?

4. METHODOLOGY

4.1 Subjects

The study presented in this article was carried out in the context of a 3rd semester course on "Object-Oriented Programming" that is taught at the Department of Applied Informatics in the University of Macedonia, Greece. Important information regarding the educational context of the course under study and related courses taught before this one are presented in Table 1. The pair programming methodology and guidelines on SCEPPSys were presented in a meeting at the beginning of the course in order to eliminate factors related to remote communication, such as the ability to use collaborative tools effectively and better understand the practices and responsibilities of each role (da Silva Estácio and Prikladnicki, 2015). In a two-hour lab the process of collaboratively solving an assignment using SCEPPSys was presented to students. After this presentation students started solving a demo assignment. Students were positioned at adjacent computers in order to be able to see their partner and experience how a DPP session works. The instructor observed students, answered questions and in the case of situations that the students considered as system problems showed them the appropriate way of using it. Students were assigned 6 Java programming assignments as homework with a deadline of 10 days. The assignments were carried out by pairs of students using the SCEPPSys Eclipse plugin for the application of DPP. Students worked on their own schedules remotely from home. Periodical tests of logged IP addresses were performed to ensure that DPP was working properly, namely that students were not co-located. Students chose their own partner using an online form. Pair members did not change throughout the course. Important information regarding the DPP assignments and their content is presented in Table 2. The final exam was carried out in the lab where students had to develop a complete program individually. The grade of each DPP assignment was

the same for both members of each pair of students and the assignments grade accounted for 15% of the final grade.

Table 1. Educational context of OOP course

Department/Institution	Applied Informatics/University of Macedonia
Course (semester)	Object-Oriented Programming (3rd semester)
Programming language	Java
Syllabus	▪ Objects and classes (necessity of using classes) ▪ Class definition (fields, constructors, methods) ▪ Constructing objects and invoking methods (main) ▪ Class associations ▪ Groups of objects (array, ArrayList) ▪ Inheritance, polymorphism and overriding ▪ Abstract classes and interfaces ▪ Graphical User Interface (constructing a simple GUI, event handling, interaction with domain classes) ▪ Collection framework of Java ▪ Manipulation of text and binary files
Duration	13 weeks, 3 hours per week
Teaching approach	▪ 3 hour lab session every week ▪ The course is taught by two instructors using the same educational material ▪ Educational material is accessible through an asychronous course management system ▪ OOP concepts are approached through hands on exercises at lab ▪ BlueJ is used for presenting the structure (simplified UML class diagram) of projects ▪ Eclipse is used for programming exercises at labs and assignments ▪ New OOP concepts are presented in the context of extending projects developed in previous lab sessions
Relevant courses	▪ Algorithms in C (1st semester) ▪ Procedural Programming in C (1st semester) ▪ Data Structures (2nd semester)
Course enrollment	Students can enroll in a course even if they have not passed prior relevant courses.

Ninety-four (94) students attended the Java course during the semester. 78% of them (73 students) participated in the final examination and thus obtained a Java exam grade. However, sixty-two (62) students have been retained for the statistical analysis as these students had also participated in the exams of the preceding programming course (i.e. had a Prior grade) and their partner in their pair also had a Prior grade (i.e. belonged to a pair that we considered as a 'healthy' case for the analysis).

The distribution of students' final grades in the prior C programming course and in the examined OOP course is illustrated in Figure 1. Grades are measured on a scale from 0 to 10, where 10 is "excellent" and minimum passing grade is 5. 65% of students passed the prior C programming course, while 47% passed the OOP course in Java. In the Department of Applied Informatics it is quite common that the pass rate for the OOP course is lower than that of the preceding C course, due to the fact that the object-oriented development paradigm is more demanding. Students, in general, face difficulties in comprehending even the fundamental concepts of objects and classes (Eckerdal & Thuné, 2005; Xinogalos, 2015), while prior experience on procedural programming is considered to make the transition to OOP more difficult.

Table 2. Educational context of DPP assignments

Academic year	2015-16
Participants	94 students attended the course 73 students participated in the final examination 62 students retained for the statistical analysis as explained
Prior programming knowledge	1st semester "Procedural Programming" course based on C 41% of the participants had either failed or dropped out the course.
Prior experience on DPP	None
Group formation	Free selection of partner
DPP system	SCEPPSys
Assignments (content)	1. Class definition, main 2. Class associations 3. Object collections – ArrayList 4. Inheritance & polymorphism 5. GUI, event handling (& inheritance) 6. Binary files (&inheritance, ArrayList, Comparator)
Deadline for each assignment	10 days

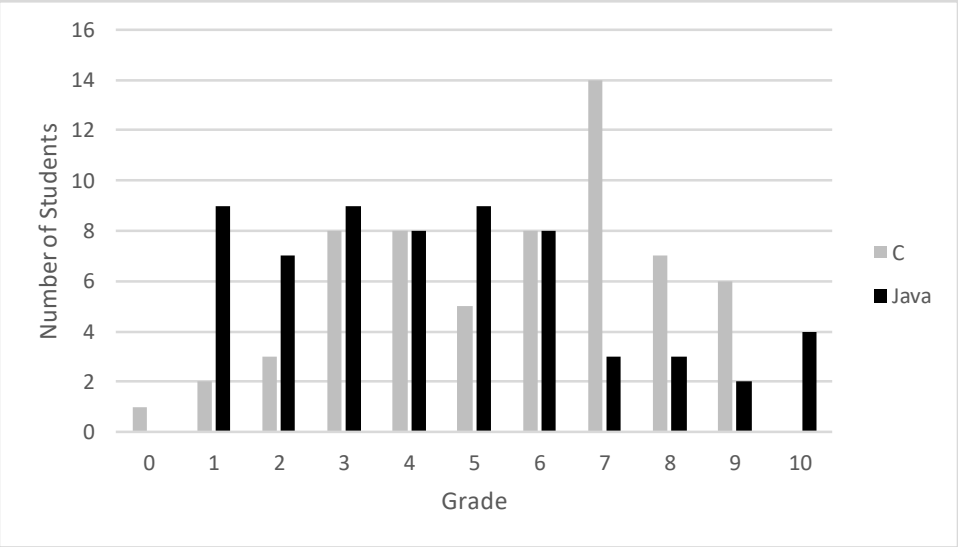

Figure 1. Distribution of student final grades in prior C course and in the examined Java course

4.2 Distributed Pair Programming Environment

The features of SCEPPSys that has been used as the DPP environment are summarized below:

SCEPPSys supports the basic requirements of remote collaboration. Pair programmers can edit the source code in real time using a shared editor which supports the roles of the driver and navigator. Code editing is allowed only for one user each time, namely for the driver. An embedded text-based communication channel allows programmers to discuss and coordinate their actions. Remote code highlighting, a basic gesturing feature, enables the navigator to point out code parts in order to indicate potential problems. The remaining features, the so-called "awareness indicators", aim to provide to pair programmers information about user's status and performed actions within the workspace (like editing, saving etc.).

DPP is practiced in the form of collaboration scripts. Scripts consist of a number of components and mechanisms (Kobbe et al., 2007). These factors were adapted to the requirements of DPP and can be defined through the administration environment of SCEPPSys. The authoring procedure of a script includes the settings of participants; pair formation based on: free selection, random selection, comparable skill levels, comparable code contribution levels; programming tasks and turn-taking policies. Programming tasks are sub-tasks of a major programming assignment. Turn-taking policies specify the distribution of the driver/navigator roles between the programmers. The teacher may select one of the following task distribution policies, or may apply different policies within one class.

(1) Free collaboration: This policy allows students to distribute tasks based on their own decisions.
(2) Rotating Roles: The system performs successive role alternations between driver and navigator.
(3) Balanced knowledge acquisition: The system assigns a task to a student based on the learning objective of the task and his/her knowledge acquisition compared to his partner's up to a particular step of the program.

Students remotely solve the assigned programming tasks using the Eclipse plugin SCEPPSys. The entire problem-solving procedure, including the problem statement of each programming task, is guided by the collaboration script prepared by the instructor. The workspace of the plugin allows the interpretation of scripts by creating student pairs, granting floor control, and displaying the problem statement of programming tasks in a separate Eclipse window.

Compared to other Eclipse plugins that support DPP, SCEPPSys contains an extended administration environment which is a web based system (Tsompanoudi and Satratzemi, 2014). It allows instructors to organize programming assignments, monitor the outcomes of students' collaboration, and assess submitted programming assignments. In order to assess each student's contribution to the submitted programming assignment, the system saves the following values: driving time (the time each member of a pair spent "driving", e.g. typing the program code) and non-driving time of each student, total time (the time each pair

spent on solving the programming assignment), amount of inserted program code per student, number of exchanged messages, number of program executions, number of role switches, distribution of tasks per student, productivity (relationship between driving time and total time), grade of each programming assignment.

For the current study the applied pair formation was free selection since students formed pairs on their own. A significant feature of SCEPPSys is its ability to distribute the roles of the driver and navigator to the pair members. This aims to promote participation and teach students how to collaborate effectively. In one of our previous studies turn-taking was fluid and students were responsible to distribute roles. However this led to disengagement on part of some students. We studied alternative turn-taking policies supported by SCEPPSys and found that rotating roles is more effective and accepted by the students. Therefore the chosen task distribution policy for this study was rotating roles, meaning that pairs switched roles after each task. The plugin contains an embedded chat but students were also allowed to use external communication channels since our previous studies showed that most pairs combine text-based and audio-based communication.

4.3 Variables

Student's actual skill is determined based on the student's prior programming performance, i.e. his/her prior performance in C language. Student prior programming performance was defined by the final grade (*Prior Grade*) in the introductory programming course taught in the 1st semester. This is an introductory procedural programming course that uses C as the programming language. In analogy to most previous studies investigating the pairing of students, student actual skill is additionally characterized by an ordinal variable. In particular, based on the *Prior Grade* students were classified in four different groups as presented in Table 3.

Table 3. Classification of Students Based on Prior Programming Performance in C

***Prior Grade* in an introductory programming course (C Language)**	***Prior Grade Level***
9 – 10	A
7 – 8	B
5 – 6	C
Fail	D

Pair actual skill was approached in two ways as follows:
Pair Prior Grade which is the mean grade of the pair members in the C course and *Pair Prior Grade Level* which is defined as in Table 4.

Table 4. Classification of Pairs Based on pair Prior Programming Performance in C

Pair Prior Grade	***Pair Prior Grade Level***
9 – 10	A
7 – 8	B
5 – 6	C
Fail	D

The transformation of *Prior Grade* and *Pair Prior Grade* to four different groups as defined in Table 3 and Table 4 respectively, was made to explore if there is a significant effect of previous programming performance of student and pair on the performance in an OOP course carrying out DPP programming assignments. We adopted the Greek grading system of higher education for the classification of students in four different groups. The correspondence between the classification of students in four different groups presented in Table 3, the Greek grading system, and ECTS is as follows:

A, Excellent, ECTS A
B, Very good, ECTS B
C, Good, ECTS C
D, Fail, -

Pair compatibility was approached employing the metric *Compatibility Degree* based on the pair's prior programming performance as follows:

Compatibility Degree was defined as the absolute difference of *Prior Grade Level* between the members of each pair, which can range from 0 to 3. In Table 5 the values of *Compatibility Degree* and the different combinations of pairs leading to these values are given. The corresponding pair types are presented as an unordered pair (X, Y), where X, Y are the values of *Prior Grade Level* for each student in the pair.

Table 5. Compatibility Degree and the Corresponding Pairs

Compatibility Degree	**0**	**1**	**2**	**3**
Corresponding pairs	(A, A) (B, B) (C, C) (D, D)	(A, B) (B, C) (C, D)	(A, C) (B, D)	(A, D)

Compatibility Degree of 0 or 1 represents pairs with the same or similar programming skills. Compatibility Degree of 2 or 3 represents pairs with important differences in programming skills. Evidence from previous studies (Katira et al., 2004; Van Toll et al. 2007; Williams et al. 2006) on compatibility and PP effectiveness implies that PP works best when pair members have a similar skill level. Based on these findings we examine if for pairs with a *Compatibility Degree* of 0 or 1 (Compatible Pairs (CP) in our case), DPP is more effective than for pairs with a *Compatibility Degree* of 2 or 3 (Non Compatible Pairs (NCP) in our case).

Mean Code Contribution for each student is calculated as the ratio of the typed code in the DPP assignments to the total code of the submitted assignments. This information is part of the data collected by SCEPPSys during the problem-solving process of each assignment.

We ran several statistical tests in order to answer the research questions. All variables used in the tests are summarized in Table 6. Each variable has a short description and an indication whether the variable refers to the student, his/her partner or the pair to which he/she belongs.

Table 6. Variables Used in Statistical Tests

Variable name	**Description**	***Applies to***		
		Student	**Partner**	**Pair**
Prior Grade	Student overall performance in the C course	+		
Prior Grade Level	Student overall performance in the C course as defined in Table 3	+		
Partner Prior Grade	*Partner*'s overall performance in the C course		+	
Partner Prior Grade Level	*Partner* overall performance in the C course as defined in Table 3		+	
Number of Submitted Assignments	The number of assignments submitted by each pair			+
Java Assignments Grade	Mean grade of submitted assignments for each pair			+
Mean Code Contribution	Mean contribution rate in the program code in each of the DPP assignments	+	+	
Java Exam Grade	Grade received in the final Java examination	+		
Pair Prior Grade	Mean grade of the pair members in the C course			+
Pair Prior Grade Level	Mean grade of the pair members in the C course as defined in Table 3			+
Compatibility Degree	Pair's compatibility degree as defined in Table 4			+

4.4 Data Analysis

As already mentioned, the purpose of this study is to examine the impact of various factors on the performance of CS students, carrying out DPP programming assignments in the context of an OOP course. The association between the investigated research questions (see Section 3) and the involved variables (see Section 4.3) is presented in Table 7.

Table 7. Data Analysis

Research Question	**Variables**	
	independent*	***dependent****
RQ1: Does the student's actual skill affect his/her performance on an OOP course that is supported by DPP assignments?	*Prior Grade* *Prior Grade Level*	*Java Exam Grade* *Java Assignments Grade* *Mean Code Contribution*
RQ2: Does the partner's actual skill affect student performance on an OOP course that is supported by DPP assignments?	*Partner Prior Grade* *Partner Prior Grade Level*	*Java Exam Grade* *Java Assignments Grade* *Mean Code Contribution*
RQ3: Does the pair's actual skill and the pair's compatibility affect student performance on an OOP course that is supported by DPP assignments?	*Pair Prior Grade* *Pair Prior Grade Level* *Compatibility Degree*	*Java Exam Grade* *Java Assignments Grade*
RQ4: Does the pair's performance on DPP assignments affect student performance on an OOP course?	*Number of Submitted Assignments* *Java Assignments Grade*	*Java Exam Grade*

*The terms 'dependent' and 'independent' for the involved variables are meaningful only in the context of ANOVA analysis.

Statistical analysis was performed by using IBM SPSS Statistics (version 19.0.0). The mean (M) and standard deviation (SD) were computed for quantitative variables. The Pearson's product-moment correlation coefficient (r) was computed to assess the relationship between quantitative variables. A one-way between subjects Analysis of Variance was conducted:

a) to compare the effect of *Prior Grade Level* on *Java Exam Grade* and on mean *Java Assignments Grade* (as part of RQ1);

b) to compare the effect of *Partner Prior Grade Level* on *Java Exam Grade* and on mean *Java Assignments Grade* (as part of RQ2);

c) to compare the effect of *Pair Prior Grade Level* on *Java Exam Grade* and on mean *Java Assignments Grade* (as part of RQ3).

We should point out that the group of students with *Prior Grade level* A has few observations; thus we combined those students with those of *Prior Grade Level* B and we formed a new group which we refer to as *Prior Grade Level* of A&B. Combining these two groups with *Prior Grade Level* A and *Prior Grade Level* B has a reasonable basis, and allows a more reliable application of ANOVA. Therefore, there were three groups of students A&B, C, D based on their *Prior Grade Level* / *Partner Prior Grade Level* / *Pair Prior Grade Level* respectively. When significant differences existed, post-hoc analysis was conducted to obtain specific information on which groups of students are significantly different from each other; the Games-Howell test was performed due to unequal sample sizes among the conditions of *Prior Grade Level*. Values of $p < 0.05$ were considered significant.

5. RESULTS

In this Section we present the results per Research Question without any further analysis, as the discussion of the statistical findings is performed in Section 6.

5.1 Research Question 1

RQ1: Does the student's actual skill affect his/her performance on an OOP course that is supported by DPP assignments?

The correlation analysis revealed that:

- *Java Exam Grade* correlates positively with *Prior Grade*, r=.609, p<.001.
- *Java Assignments Grade* correlates positively with *Prior Grade*, r=.501, p<.001.
- *Mean Code Contribution* does not correlate with *Prior Grade*.

The one-way ANOVA revealed that:

- There was a significant effect of *Prior Grade Level* on *Java Exam Grade* $F(2,59)=17.290$, $p<0.001$. Post-hoc tests indicated that the mean *Java Exam Grade* of students with *Prior Grade Level* of **D** (M=1.51, SD=1.20) was significantly different from the mean *Java Exam Grade* of students with *Prior Grade Level* of **A&B** (M=4.92, SD=2.66), $p<0.001$ and of students with *Prior Grade Level* of **C** (M=2.94, SD=1.59), p=0.027. Accordingly, there was a significant difference between the students with *Prior Grade Level* of **A&B** (M=4.92, SD=2.66), $p<0.001$ and the students with *Prior Grade Level* of **C** (M=2.94, SD=1.59), p=0.016 (see the corresponding descriptive statistics in Table 8). Therefore,

as conclusion we can state that students in the D *Prior Grade Level*, i.e. students who failed in a prior programming course, received significantly lower grades in Java than students in the A&B *Prior Grade Level* and also than students in the C *Prior Grade Level*. Students in the C *Prior Grade Level* received significantly lower grades in Java than students in the A&B *Prior Grade Level*.

- There was a significant effect of *Prior Grade Level* on *Java Assignments Grade* F(2,59)=6.710, p=0.002. Post-hoc tests indicated that the mean *Java Assignments Grade* of students with *Prior Grade Level* of **D** (M=6.18, SD=3.04) was significantly different from the mean *Java Assignments Grade* of students with *Prior Grade Level* of **A&B** (M=8.74, SD=1.94), p=0.005 (see the corresponding descriptive statistics in Table 8). Therefore, as conclusion we can state that students in the D *Prior Grade Level*, i.e. students who failed in a prior programming course, received significantly lower grades in Java assignments than students in the A&B *Prior Grade Level*. The mean grade in Java assignments between the other categories of *Prior Grade Level* wasn't significantly different.

Table 8. Descriptive Statistics for RQ1

					95% Confidence Interval	
	Prior Grade Level	**N**	**Mean**	**SD**	**Lower Bound**	**Upper Bound**
Java Exam Grade	D (Fail)	22	1.51	1.20	0.97	2.03
	C (5-6)	13	2.94	1.59	1.98	3.90
	B (7- 8) & A (9-10)	27	4.92	2.66	3.86	5.97
	Total	62	3.29	2.52	2.65	3.93
Java Assignments Grade	D (Fail)	22	6.18	3.04	4.83	7.53
	C (5-6)	13	7.90	2.26	6.53	9.26
	B (7- 8) & A (9-10)	27	8.74	1.94	7.97	9.51
	Total	62	7.66	2.67	6.98	8.33

5.2 Research Question 2

RQ2: Does the partner's actual skill affect student performance on an OOP course that is supported by DPP assignments?

Correlation analysis revealed that:

- *Java Exam Grade* correlates positively with *Partner Prior Grade*, r=.293, p=.021.
- *Java Assignments Grade* correlates positively with *Partner Prior Grade*, r = .428, p=.001.
- *Mean Code Contribution* does not correlate with *Partner Prior Grade*.

The one-way ANOVA revealed that:

- There was a significant effect of *Partner Prior Grade Level* on *Java Exam Grade* F(2,59)=6.193, p=0.004. Post-hoc tests indicated that the mean *Java Exam Grade* of students with *Partner Prior Grade Level* of **D** (M=1.87, SD=1.75) was significantly different from the mean *Java Exam Grade* of students with *Partner Prior Grade Level* of **A&B** (M=3.81, SD=2.29), p=0.006 and of students with *Partner Prior Grade Level* of **C** (M=4.38, SD=3.02), p=0.023 (see the corresponding descriptive statistics in Table 9). Therefore, as conclusion we can state that students with a partner in the D *Prior Grade Level* received significantly lower grades in Java than students with a partner in the A&B *Prior Grade Level* and also than students with a partner in the C *Prior Grade Level*.
- There was a significant effect of *Partner Prior Grade Level* on *Java Assignments Grade* F(2,59)=5.090, p=0.009. Post-hoc tests indicated that the mean *Java Assignments Grade* of students with *Partner Prior Grade Level* of **D** (M=6.35, SD=2.98) was significantly different from the mean *Java Assignments Grade* of students with *Partner Prior Grade Level* of **A&B** (M=8.69, SD=1.77), p=0.009 (see the corresponding descriptive statistics in Table 9). Therefore, as conclusion we can state that students with a partner in the D *Prior Grade Level* received significantly lower grades in Java assignments than students with a partner in the A&B *Prior Grade Level*.

Table 9. Descriptive Statistics for RQ2

					95% Confidence Interval	
	Partner Prior Grade Level	**N**	**Mean**	**SD**	**Lower Bound**	**Upper Bound**
Java Exam Grade	D (Fail)	21	1.87	1.75	1.07	2.67
	C (5-6)	15	4.38	3.02	2.71	6.06
	B (7- 8) & A (9-10)	26	3.81	2.29	2.88	4.73
	Total	62	3.29	2.52	2.65	3.93
Java Assignments Grade	D (Fail)	21	6.35	2.98	4.99	7.71
	C (5-6)	15	7.69	2.85	6.11	9.27
	B (7- 8) & A (9-10)	26	8.69	1.77	7.98	9.41
	Total	62	7.66	2.67	6.98	8.33

5.3 Research Question 3

RQ3: Does the pair's actual skill and the pair's compatibility affect student performance on an OOP course that is supported by DPP assignments?

Correlation analysis revealed that:

- *Pair Prior Grade* correlates positively with *Java Exam Grade*, r=.502, p<.001, and *Java Assignments Grade* r=.530, p<.001.

while the one-way ANOVA revealed that:

- There was a significant effect of *Pair Prior Grade Level* on *Java Exam Grade* F(2.59)=8.936, p<0.001. Post-hoc tests indicated that the mean *Java Exam Grade* of students with *Pair Prior Grade Level* of **D** (M=1.57, SD=1.24) was significantly different from the mean *Java Exam Grade* of students with *Pair Prior Grade Level* of **A&B** (M=4.47, SD=2.83), p<0.001 and of students with *Pair Prior Grade Level* of **C** (M=2.91, SD=1.78), p=0.044 (see the corresponding descriptive statistics in Table 10). Therefore, as conclusion we can state that pairs in the D *Pair Prior Grade Level* received significantly lower grades in Java than pairs in the A&B *Pair Prior Grade Level* and also than pairs in the C *Pair Prior Grade Level*.
- There was a significant effect of *Pair Prior Grade Level* on *Java Assignments Grade* F(2.59)=11.118, p<0.001. Post-hoc tests indicated that the mean *Java Assignments Grade* of students with *Pair Prior Grade Level* of **D** (M=5.46, SD=2.81) was significantly different from the mean *Java Assignments Grade* of students with *Pair Prior Grade Level* of **A&B** (M=8.85, SD=1.17), p=0.001 (see the corresponding descriptive statistics in Table 10). Therefore, as conclusion we can state that pairs in the D *Pair Prior Grade Level*, i.e. both members of a pair failed in a prior programming course, received significantly lower grades in Java assignments than pairs in the A&B *Pair Prior Grade Level*.

Table 10. Descriptive Statistics for RQ3 (pair's actual skill)

					95% Confidence Interval	
	Pair Prior Grade Level	**N**	**Mean**	**SD**	**Lower Bound**	**Upper Bound**
Java Exam Grade	D (Fail)	16	1.57	1.24	0.91	2.23
	C (5-6)	17	2.91	1.78	2.00	3.82
	B (7- 8) & A (9-10)	29	4.47	2.83	3.39	5.54
	Total	62	3.29	2.52	2.65	3.93
Java Assignments Grade	D (Fail)	16	5.46	2.81	3.96	6.96
	C (5-6)	17	7.69	3.15	6.07	9.30
	B (7- 8) & A (9-10)	29	8.85	1.17	8.41	9.29
	Total	62	7.66	2.67	6.98	8.33

Regarding the correlation between Compatibility Degree and the two performance indicators the analysis has revealed that:

- *Compatibility Degree* does not correlate significantly with *Java Exam Grade* or *Java Assignments Grade*.

Table 11 provides more details about the performance (average grade in Java) of students, according to their *Prior Grade Level* and the compatibility of the pair that they belong to. For example, for students belonging to a pair with *Compatibility Degree* 1, the mean *Java Grade* was 8.67 (for students having a *Prior Grade Level* of **A**) and 3.33 (for students having a *Prior Grade Level* of **D**). The far right column indicates for example that the mean *Java Grade* for all students belonging to pairs with *Compatibility Degree* 1 was 5.53.

5.4 Research Question 4

RQ4: Does the pair's performance on DPP assignments affect student performance on an OOP course?
The statistical analysis of the results revealed that:

- *Java Exam Grade* correlates positively with *Java Assignments Grade*, r=.378, p=.002.
- *Java Exam Grade* does not correlate with *Number of Submitted Assignments*.

However, we must note that the majority of students (more than 3 out of 4 students) submitted nearly all projects. The number of students who submitted at least the designated number of projects is presented in Table 12.

6. DISCUSSION

In this section, we summarize and discuss the results regarding the factors that are related to student performance in an Object-Oriented programming course, where students completed 6 Java programming assignments using the SCEPPSys DPP Eclipse plugin. The factors considered concerned each student individually, as well as pairs in terms of compatibility.

The findings to the research questions that have been posed in this study are as follows:

The student's actual skill is associated to his/her performance in an OOP course that is supported by DPP assignments. According to the analysis of RQ1, there is a statistically significant correlation between his/her overall performance in Java (strong correlation for *Java Exam Grade* and moderate correlation for *Java Assignments Grade*) and his/her previous programming performance (*Student Prior Grade*). Moreover, analysis of variance revealed a positive effect of student actual skill on his/her performance in an OOP course. More specifically, students with a lower performance in C scored less in Java exams and in Java assignments than students with a higher performance in C. This is an expected finding, regardless of whether a pair programming approach is employed. However, it should be borne in mind when designing PP and DPP assignments since prior knowledge of each pair member is a determinant factor for his learning efficiency. Moreover, this result gives the necessary evidence for persuading students that –although a department might not have any prerequisites for enrolling a course (as is the case in our department)– they should have a deep knowledge on fundamental programming concepts/constructs before enrolling in an OOP course. Especially the students who failed the prior course on C, exhibit significantly lower performance than their peers who achieved better than 70% in the C course. It should be mentioned however, that there is no statistical correlation between the contribution of each student to the amount of the pair's code and his actual skill. This could be attributed to the willingness of students to participate in distributed pair programming tasks, regardless of their skill, something that can be seen as a positive contribution of DPP to a programming course.

The partner's actual skill (as obtained by the *Prior* grade of the partner) is related to some extent to the student's performance in an OOP course that is supported by DPP assignments (RQ2), since there is a correlation between his/her performance in Java (weak correlation for *Java Exam Grade* and moderate correlation for *Java Assignments Grade*) and his/her partner's actual skill. Analysis of variance revealed a positive effect of *Partner Prior Grade Level* on the overall performance of a student in Java (as recorded by variables *Java Exam Grade and Java Assignments Grade*). This implies that the transfer of knowledge that has been recorded for PP (Faja, 2011) is achieved in the context of DPP assignments as well. This finding renders even more important the criteria for forming pairs since learning is related not only to the student's own effort, but depends on his partner's skills as well, even to a weak/moderate degree.

Table 11. Descriptive Statistics for RQ3 (Compatibility Degree)

	Prior Grade Level	A				B				C				D				Total			
Compatibility Degree	**# of students**	**M**	**SD**	**# pass**	**# fail**	**M**	**SD**	**# pass**	**# fail**	**M**	**SD**	**# pass**	**# fail**	**M**	**SD**	**# pass**	**# fail**	**M**	**SD**	**# pass**	**# fail**
0	33					5.50	2.15	9	3	4.43	1.81	4	3	2.29	1.20	0	14	3.91	2.23	13	20
1	17	8.67	1.16	3	0	6.50	2.88	5	1	3.80	1.79	2	3	3.33	1.16	0	3	5.53	2.79	10	7
2	9	8.00	2.83	2	0	5.00	4.00	2	1	7.00	-	1	0	4.00	1.00	1	2	5.56	2.83	6	3
3	3	3.00	-	0	1									1.00	0.00	0	2	1.67	1.16	0	3
Total	**62**	**7.50**	**2.67**	**5**	**1**	**5.71**	**2.55**	**16**	**5**	**4.38**	**1.85**	**7**	**6**	**2.55**	**1.34**	**1**	**21**	**4.48**	**2.60**	**29**	**33**

*A blank cell implies that no student of the corresponding Prior Grade Level belonged to a pair with the designated compatibility degree
** A '-' implies that standard deviation cannot be calculated due to a single value

Table 12. Distribution of assignment submissions

Number of submitted projects	***Number of students (percentage)***
1-6	41 (66.1%)
1-5	48 (77.4%)
1-4	54 (87.1%)
1-3	56 (90.3%)
1-2	62 (100%)

It should be noted that student performance (reflected both in the *Java Exam Grade* and the *Java Assignments Grade*) is related primarily to his own actual skill (*Prior Grade*) and secondarily to the actual skill of his partner (*Partner Prior Grade*). Student performance (*Java Exam Grade*) presents a statistically significant difference between students of which their actual skill level (*Prior Grade Level*) differs by two or more levels (e.g. *Prior Grade Levels of A and C, or A and D*). This performance disparity is less observable when the difference in the actual skill level of the partner is considered.

With respect to the pair's actual skill representing the prior knowledge of both students, a clear impact on performance of each student in an OOP course (RQ3) has been observed. The mean value of pair performance in the prior programming course (variable *Pair Prior Grade*) has a moderate positive correlation to both aspects of a student's performance, namely the *Java Exam Grade*, and the *Java Assignments Grade*. Moreover, the one-way ANOVA showed that there is an effect of variable *Pair Prior Grade Level* on both the *Java Exam Grade* and the *Java Assignments Grade*. Students belonging to stronger pairs achieve a better academic performance in the OOP course than students belonging to weaker pairs. However, the *Compatibility Degree* has not yielded any statistically significant results. These results indicate that the measure of pair compatibility regarding pair's prior programming skills cannot permit us to reason about student performance in an OOP course that is supported by DPP assignments. The number of different groups is very small in categories with *Compatibility Degree* 2 or 3 (Table 11) i.e. pairs with important differences in their programming skills, and thus we could not make any significant deductions. The results of pairs with *Compatibility Degree* 0 or 1 (Table 11), i.e. compatible pairs regarding their prior programming skills, could give us some indication as to how this factor is associated with students' performance in an OOP course. Pairs of types (A, B) or (B, C) attained similar performance as they did in a previous programming course. Students with weak prior programming performance, i.e. D level students, when collaborating in DPP assignments with C level students achieved a better performance in the OOP course than those who collaborated with D level students. This is an indication that when both members of a pair have poor programming skills they are not able to benefit from DPP sessions, but when only one member has weaker programming skills it appears that she/he might benefit from DPP sessions when collaborating with better students. Even though *Compatibility Degree* expresses the similarity or the difference of prior programming skills between the two members of a pair it could not be surmised whether or not this factor had a positive effect on each member's performance in DPP. Additional qualitative data from participants can help to rule out special cases. A questionnaire or interviews with pairs with important differences in their programming skills about their perceptions on compatibility and DPP effectiveness could might help to shed light on those cases. As mentioned in the related work section the relevant studies (Katira et al., 2004; Van Toll et al. 2007; Williams et al. 2006) on compatibility and PP effectiveness have shown that PP works best when the pair has a similar skill level. In these studies compatibility was expressed in a subjective way. Taking into account this observation our work can be extended by collecting additional data about the perception of students on their partner compatibility concerning partner skill level and DPP effectiveness.

Finally, with respect to DPP assignments as a means of improving learning performance the results indicate that pair performance on DPP assignments is related, even to a small extent, to student performance in an OOP course (RQ4). There is a statistically significant weak correlation between *Java Assignments Grade* and *Java Exam Grade* (however, it should be noted that student performance does not show any difference depending on the number of submitted assignments). Such findings confirm the belief that active engagement of students in programming assignments and pair programming in particular contribute to the improvement of their academic performance. The fact that this holds also for distributed pair programming assignments facilitates computer science department to adopt technology furthermore in their curricula without deteriorating the quality of studies.

7. THREATS TO VALIDITY

In this section, we present and discuss threats to the validity of the empirical study emphasizing on construct, reliability, external and internal validity threats, according to the classification by Runeson et al. (2012).

Construct validity reflects to what extent the phenomenon under study (i.e. the relation of skill level and compatibility to student performance in the context of an OOP course supported by DPP assignments) really represents what is investigated according to the research questions. By selecting particular metrics for quantifying skill level, compatibility and performance, threats to construct validity emerge. For example, student performance is assessed employing his/her grade (in the final exam or the assignments), whereas

performance might also be improved in terms of number of compile- or run-time errors or even in terms of code quality. Although grades, especially in a programming course, reflect only some aspects of the learning process, we have relied on them in analogy to most studies, where summative assessment is carried out to evaluate student learning. Furthermore, as a mitigation action we opted for assessing performance by multiple indicators, namely *Java Assignments Grade*, *Mean Code Contribution*, *Java Exam Grade*. Another threat to construct validity is related to the fact that students were able to choose their own partners, possibly resulting in an imbalance in the number of pair formation types as based on the *Compatibility Degree* and the *Pair Prior Grade Level* variables (used in the one-way ANOVA analysis). However, as it can be observed from Table 10 pairs of students of all kinds have been formed (except for pairs with both students in the highest skill level), partially mitigating the aforementioned threat.

The reliability of a case study is related to the extent by which the collected information and the performed analysis can be replicated with the same results. To mitigate reliability threats we explicitly report the design of the case study and the statistical tests.

Internal validity threats are related to the identification of confounding factors, that is, variables, other than the selected independent variables (*Pair Prior Grade Level, Prior Grade Level, and Partner Prior Grade Level*) which might influence the value of the dependent variable (performance in the context of an OOP course). Such threats do apply in the presented study, since the performance of a student who participated in DPP assignments have also been affected by factors beyond our control or ability to measure them. It is reasonable to assume that performance in final exam is affected by other parameters such as study time, intelligence and general background knowledge, which are excluded from our study. To limit the impact of such latent variables, performance indicators include beyond the *Java Exam Grade*, the *Java Assignments Grade* and *Mean Code Contribution* as well. Moreover, the performance of students in distributed teams might have been affected by factors other than their actual skill such as their prior exposure to working in teams or the ability to effectively use collaborative tools. To mitigate this threat the employed collaborative environment SCEPPSys has been presented in a two-hour meeting at the beginning of the course and after that students were invited to solve a demo assignment and questions have been discussed.

Finally, as in any other empirical study, the results are subject to external validity threats. External validity deals with our possibility to generalize the findings regarding the relation between skill/compatibility and performance in the context of DPP, beyond the students of this study and to other course/programming languages. It is reasonable to assume that individual findings might differ for other programming languages or students with different prior skills and even more if DPP is applied in an introductory programming course. However, most results can be considered intuitive from an educational perspective and the statistical analysis provides arguments in favor of DPP as an instrument to support individual learning activities in programming courses.

8. CONCLUSION

The goal of this study was to investigate whether prior programming skills assessed at the level of student, his/her partner, of the pair as a whole and pair compatibility are related to student performance in an OOP course that is supported by DPP assignments. To further examine the role of DPP on learning outcomes we have studied whether pair's performance in DPP assignments is related to students' grade. Although similar studies can be found for assignments carried out by pair programming, to the best of our knowledge no prior studies have examined the influence of the aforementioned factors on the effectiveness of DPP.

The study showed that it is not just the student's prior programming skill that is associated to his/her performance in an OOP course –as expected-, but also the partner's prior programming skill. The findings indicate that the student's actual skill, as well as the pair's actual skill both have a positive effect on the performance of the student in an OOP course. Student performance in the collaborative solution of DPP assignments has a relationship with the performance in the exams.

Overall through our research, we found that pairs of students with a collectively better performance in the prior C programming course maintain a good academic performance, while pairs with a collectively poor performance in the prior C programming course have a substantially lower performance in Java. However, the difference in the programming skills between the members of a pair, expressed as their compatibility degree, does not seem to be related to his/her performance.

Our findings seem to confirm partially with the results reported only by Katira et al. (2004) on PP concerning pair compatibility. Their results suggested that there is no association between the actual skill level and the compatibility for the two of the three of the studies they conducted. However, so as to

generalize our findings, more empirical studies need to be conducted on pair compatibility and students' performance by having the same number of pair formation types based on skill level. Moreover, it would be valuable to replicate previous studies to assess whether DPP is effective on student performance compared to the performance of students not involved in pair programming assignments. Furthermore, such studies should focus on the improvement of programming skills not necessarily reflected in the final grade of a course.

The findings of our research can be used by instructors to help them understand how to implement DPP in CS courses. The fact that the choice of the level of student pairs based on students' prior performance in a similar programming course is related to their performance means that instructors should avoid the formation of pairs with both students having weak actual skills, as students belonging to such pairs will not benefit from the collaboration. When students form freely pairs, as was the case in our study, they should be informed that they should take into account their prior performance and not just friendship relationships.

REFERENCES

Baheti, P., Gehringer, E., & Stotts, D. (2002). Exploring the efficacy of distributed pair programming.Extreme Programming and Agile MethodsXP/Agile Universe, 11(1), 2.

Braught, G., Wahls, T., & Eby, L. M. (2011). The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education (TOCE)*, *11*(1), 2.

Canfora, G., Cimitile, A., Garcia, F., Piattini, M., & Visaggio, C. A. (2005). Confirming the influence of educational background in pair-design knowledge through experiments. In *Proceedings of the 2005 ACM symposium on Applied computing*, 1478-1484.

Canfora, G., Cimitle, A., & Visaggio, C. (2003). Lessons learned about distributed pair programming: what the knowledge needs to address? In: Proceedings of the Twelfth IEEE International Conference of Enabling Technologies; Infrastructure for Collaborative Enterprises, 314–319.

Cliburn, D. C. (2003). Experiences with pair programming at a small college. *Journal of Computing Sciences in Colleges*, *19*(1), 20-29.

da Silva Estácio, B. J., & Prikladnicki, R. (2015). Distributed pair programming: A systematic literature review. *Information and Software Technology*, *63*, 1-10.

Duque, R., Bravo, C. (2008). Analyzing work productivity and program quality in collaborative programming. In: Proceedings of the Third International Conference on Software Engineering Advances (ICSEA '08), IEEE Computer Society, Washington, DC, USA, 270–276.

Eckerdal, A., & Thuné, M. (2005). Novice Java programmers' conceptions of "object" and "class", and variation theory. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE'05). ACM, New York, NY, 89-93. DOI:http://doi.acm.org/10.1145/1067445.1067473

Faja, S. (2011). Pair programming as a team based learning activity: a review of research. *Issues in Information Systems*, *12*(2), 207-216.

Hanks, B. (2008). Empirical evaluation of distributed pair programming. *International Journal of Human-Computer Studies*, *66*(7), 530-544.

Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., & Zander, C. (2011). Pair programming in education: a literature review. *Computer Science Education*, *21*(2), 135-173.

Katira, N., Williams, L., Wiebe, E., Miller, C., Balik, S., & Gehringer, E. (2004). On understanding compatibility of student pair programmers. *ACM SIGCSE Bulletin*, Vol. 36, No. 1, 7-11.

Kobbe, L., Weinberger, A., & Dillenbourg, P., Harrer, A., Hämäläinen, R., Häkkinen, P., & Fischer, F. (2007). Specifying computer-supported collaboration scripts. *International Journal of Computer-Supported Collaborative Learning*, *2*(2-3), 211-224.

McDowell, C., Hanks, B., & Werner, L. (2003). Experimenting with pair programming in the classroom. In *ACM SIGCSE Bulletin* ,Vol. 35, No. 3, 60-64. ACM.

Mendes, E., Al-Fakhri, L., & Luxton-Reilly, A. (2006). A replicated experiment of pair-programming in a 2nd-year software development and design computer science course. *ACM SIGCSE Bulletin*, *38*(3), 108-112.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003). Improving the CS1 experience with pair programming. *ACM SIGCSE Bulletin*, *35*(1), 359-362.

Plonka, L., Sharp, H., & van der Linden, J. (2012). Disengagement in pair programming: Does it matter? 34th International Conference on Software Engineering (ICSE), 496–506.

Rosen, E., Salinger, S., & Oezbek, C. (2010). Project Kick-off with Distributed Pair Programming. In *Workshop of Psychology of Programming Interest Group, Madrid*.
Runeson, P., Ho¨st, M., Rainer, A., & Regnell, B. (2012).Case study research in software engineering: Guidelines and examples. Hoboken, New Jersey: John Wiley & Sons.
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.
Schenk, J., Prechelt, L., & Salinger, S. (2014). Distributed-Pair Programming can work well and is not just Distributed Pair-Programming. *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM.
Schümmer, T., & Lukosch, S. (2009). Understanding Tools and Practices for Distributed Pair Programming. Journal of Universal Computer Science, 15 (16), 3101-3125.
Sfetsos, P., Adamidis, P., Angelis, L., Stamelos, I., & Deligiannis, I. (2012). Investigating the Impact of Personality and Temperament Traits on Pair Programming: A Controlled Experiment Replication. In Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the (pp. 57-65). IEEE
Stotts, D., Williams, L., & Nagappan, N., Baheti, P., Jen, D., & Jackson, A. (2003). Virtual teaming: Experiments and experiences with distributed pair programming. In *Conference on Extreme Programming and Agile Methods* (pp. 129-141). Springer Berlin Heidelberg.
Tsompanoudi, D., & Satratzemi, M. (2014). A Web-based authoring tool for scripting distributed pair programming. In *2014 IEEE 14th International Conference on Advanced Learning Technologies* (pp. 259-263). IEEE.
Tsompanoudi, D., Satratzemi, M., & Xinogalos, S. (2013). Exploring the effects of collaboration scripts embedded in a distributed pair programming system. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, 225-230.
Tsompanoudi, D., Satratzemi, M., & Xinogalos, S. (2015). Distributed Pair Programming Using Collaboration Scripts: An Educational System and Initial Results. *Informatics in Education*, *14*(2), 291.
Tsompanoudi, D., Satratzemi, M., & Xinogalos, S. (2016). Evaluating the Effects of Scripted Distributed Pair Programming on Student Performance and Participation. *IEEE Transactions on Education*, *59*(1), 24-31.
Van Toll, T, Lee, R., & Ahlswede, T. (2007). Evaluating the usefulness of pair programming in a classroom setting. In *6th IEEE/ACIS International Conference on Computer and Information Science* (ICIS 2007), 302-308.
Williams, L. A. (2007). Lessons learned from seven years of pair programming at North Carolina State University. *ACM SIGCSE Bulletin*, *39*(4), 79-83.
Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, J. (2000). Strengthening the case for pair programming. *IEEE Software*, *17*(4), 19.
Williams, L., Layman, L., Osborne, J., & Katira, N. (2006). Examining the compatibility of student pair programmers. In *AGILE 2006 (AGILE'06),* (pp. 10-pp). IEEE.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2000). Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers.
Xinogalos, S. (2015). Object-Oriented Design and Programming: An Investigation of Novices' Conceptions on Objects and Classes. *Trans. Comput. Educ.* 15, 3, Article 13 (July 2015), 21 pages. DOI=http://dx.doi.org/10.1145/2700519
Zacharis, N. Z. (2011). Measuring the effects of virtual pair programming in an introductory programming java course. *IEEE Transactions on Education*, *54*(1), 168-170.