Evaluating the Effects of Scripted Distributed Pair Programming on Student Performance and Participation

Despina Tsompanoudi, Maya Satratzemi and Stelios Xinogalos

***Abstract*—The results presented in this paper contribute to research on two different areas of teaching methods: Distributed Pair Programming (DPP) and Computer-Supported Collaborative Learning (CSCL). An evaluation study of a DPP system that supports collaboration scripts was conducted over one semester of a computer science course. Seventy-four students participated in the study and used the DPP model to solve programming assignments in pairs, rather than individually. Students were divided into two cohorts in order to examine how best to distribute programming tasks to students, to maximize learning outcomes. The results suggest that while the use of DPP improves pass rates and gets positive feedback from students, the use of collaboration scripts yields equal task contributions from each of the student pair. For a small number of students, overall performance was improved by the adaptive assignment of programming activities, designed to engage students equally in diverse roles and activities.**

***Index Terms*—Adaptive collaboration support, collaboration scripts, collaborative programming, distributed pair programming.**

I. Introduction

DISTRIBUTED Pair Programming (DPP) and Computer Supported Collaborative Learning (CSCL) are considered as separate research areas, but both involve forms of collaborative learning. DPP involves shared problem-solving, with two programmers working remotely to develop software. This constitutes an alternative to co-located pair programming, whose use in computer science courses has been studied for several years, with positive findings on students' performance and attitude [1]-[3]. Although fewer studies have examined DPP in the classroom, these indicate that DPP achieves most of the benefits of Pair Programming (PP) [4], [5]. The only difference between DPP and PP is DPP's allowing geographically-distributed teams to collaborate and share program code. However, such collaboration is only feasible if an underlying infrastructure enables for all necessary interactions.

A considerable number of solutions used to implement DPP were built as plugins for the Eclipse IDE. These plugins support DPP for the Java programming language, within a very popular development environment. A detailed review of all available DPP solutions reveals that most tools lack effective instructional or student support [6]. Although they cover the basic requirements of DPP, they cannot address common problems of PP, such as unequal contributions from each member of the student pair. The study reported here included the design of the SCEPPSys system (Scripted Collaboration in an Educational Pair Programming System), a more education-oriented system for using DPP in the classroom. Compared to other plugins like Sangam [7], RIPPLE [8] and XPairtise [9], SCEPPSys saves and analyzes users' interactions, and helps educators in organizing and monitoring DPP classes. Its design may also facilitate the process of DPP evaluations. So far the system has been used in two evaluation studies. Building a system with assessment and evaluation capabilities can provide the basis for further research, since extensive studies of DPP plugins are still lacking. For instance, RIPPLE was evaluated on a laboratory assignment in two user tests [8]. Students reported that they found the system easy to use, enjoyed the lab assignment, and would use it again if given the opportunity. XPairtise was evaluated over an 18-week period [9], with the focus on studying student interactions, emphasizing users' contributions and communications. These evaluations did not study the impact of DPP on student performance or pass rates on computer science courses.

Previous work in the field includes a pilot study and an evaluation study with student volunteers, using a SCEPPSys prototype; the study results were used to improve the system with new functionalities. The pilot study, which took one laboratory hour, was to test the consistency of the system and the effects of DPP using collaboration scripts [10]. Ten randomly-formed groups used the system to solve a programming assignment in Java, and switched roles according to a "scripted roles" policy. The students were quite positive about the overall experience, but had unequal participation rates and each individual changed roles as they felt like it. The pilot study also helped to detect a system bug. Another evaluation, longer and with a larger sample size, was undertaken in the spring semester of 2013, to study students' participation rates over a series of programming assignments. The goal of this eight-week study was to test the system when students worked remotely from home, and to compare scripted collaboration with free collaboration. Forty-eight student volunteers used the system to solve five Java projects. Their work was not part of the course curriculum, but students' participation earned them extra credit for another course. The

study results revealed that scripted collaboration led students to perform more role switches, and to contribute equally to the development process. Nevertheless, this did not constitute a study of the effects of DPP in authentic learning conditions.

The study reported here complements this previous work, in that it was performed as part of a typical Java course. A significant contribution of this paper is a report on the impact of DPP in authentic learning situations. For this study a Web-based administration environment was developed, instead of the former Eclipse-based solution, to make system administration more functional for the instructor [11]. As in the previous studies, SCEPPSys was used to conduct the evaluation phase, which was based on quantitative and qualitative metrics.

As well as the DPP evaluation results, other findings may be of interest to the CSCL-system research community. A core function of SCEPPSys is to support collaboration scripts for PP. Collaboration scripts are widely used in collaborative learning activities because they provide a way to structure learners' interactions. The script describes the most important aspects of a collaborative activity, such as information about participants, roles, tasks, groups and resources, and instructions on group formation, task distribution and task sequencing [12]. Previous PP and DPP studies report asymmetries in student participation levels [9], [13]. Particularly in DPP, the evaluation of XPairtise revealed either an absence of, or few, role switches in 52 DPP sessions observed. The methodology proposed combines collaboration scripts and DPP to address unequal student participation, a common problem of group work.

To achieve frequent role switches and comparable student contributions, two different approaches were explored in the part of the collaboration script responsible for distributing tasks between group members. In the first approach, users switched roles after each task, while in the second approach role assignments depended on the type of task and users' personal skills. This form of turn-taking support is described in the CSCL field as "scripted roles", whose goal is to engage students equally in relevant roles and activities [14]. In contrast to spontaneous role assignments, scripted roles are defined by the instructor and are equally distributed between group members [14]. In the current study it is reported how scripted roles were applied in the context of DPP and how they affected students' engagement and knowledge building. Role assignments were automatically performed by the system, but students were also given the possibility of switching roles on their own, so as to avoid the risk of "over-scripting" [15]. A script should shape the collaborative learning process, but without hindering natural interactions.

The next section provides an overview of the system used in the study. Sections III and IV describe the collaboration script, and the experimental design, respectively. The evaluation results are presented in Section V, followed by a brief discussion in Section VI. Conclusions and future directions are discussed in the Section VII.

II. The SCEPPSys System

The SCEPPSys system that supports the application of DPP consists of an Eclipse plugin, Fig. 1, and a Web-based administration environment [11]. The participating students used the plugin to solve a number of programming assignments. As the PP methodology suggests, the plugin supports the roles of the "driver", responsible for typing the code into the shared editor, Fig. 1(a), and the "navigator", responsible for reviewing the code.

The plugin's main features, designed to support the application of DPP, include an embedded chat tool, Fig. 1(b), remote code highlighting, and system-driven or user-driven role changes. Basic awareness indicators are used to show user status, Fig. 1(d), and assigned roles, Fig. 1(e). Programming assignments are displayed inside the Eclipse workspace, and a collaboration script guides student pairs through a sequence of tasks to the solution of a programming assignment. The script instructions for each task's problem statement are displayed in a separate Eclipse view, Fig. 1(c).

Preparing collaboration scripts using the system involves defining a number of components and mechanisms. Each script includes the participants, groups, programming tasks, and rules for task distribution and task sequencing. The collaboration scripts used in the study were prepared in the administration environment by the instructor. The organization of the DPP groups was also performed in the system, which allows the creation of user accounts and groups, and supports various group formation strategies. The project preparation, scheduling and assessment processes also form part of the administration tools. Additionally, for each project the system calculates statistical information at both the user and group level, which was used during the evaluation phase. A summary of the main findings are presented here.

III. The Script

The participants were assigned programming assignments in Java and were asked to use the SCEPPSys plugin to pair program and submit their projects. Group formation was based on the students' programming grades in previous courses, in order to create groups with comparable skill levels. This group formation strategy was adopted because previous studies reported that it had positive effects on motivation and performance [16], [17]. During DPP sessions, students were automatically assigned the roles of the "driver" and the "navigator" according to the task distribution policy chosen by the instructor. In half of the groups, the system performed successive role alternations: pairs switched roles after each task. For the other groups, the task distribution policy was based on the learning objective of the activity and students' individual skills. Each task's learning objective was defined by the instructor in the administration tool while preparing the assignment. Students' individual skills for each learning objective were estimated from their grades in previous activities that had been assessed with the help of the system. For each individual activity, the task distribution mechanism assigned the driver role to the student who had least

experience in the area of that particular learning objective. To do this, it considered how many similar activities the student had engaged it, and the overall score achieved. Consequently, by the end of the semester each student should have acquired the same level of experience for each learning objective. When the task distribution mechanism detected an equal amount of solved activities for each student, it considered each student's overall score. The goal of this policy was to achieve symmetry in skill acquisition, given students' tendency to distribute group work based on their abilities and personal needs [14], [18]. Therefore, in order to maximize the learning potential, roles were adaptively distributed to deviate from personal skills [14]. Implementing this approach of providing comparable learning opportunities to each student should not come at the cost of equal participation. Even if a student pair had similar skill levels, the expected benefit was to engage each in all of the Java concepts.

Although the roles were assigned in two different ways, all the student groups solved the same Java programming assignments and followed the same script, to create two study conditions.

IV. Experimental Design

The evaluation phase was conducted in the spring semester of 2014 in the Technology Management direction of the Department of Applied Informatics at the University of Macedonia, Greece, as part of the course Object-Oriented Design and Programming. This second-year course offers an introduction to the Java programming language and is part of the curriculum. Thirteen weeks long, it has a one-hour lecture and a two-hour lab class each week. Students attending this course earn 20% of their final grade by solving eight Java projects during the semester, 10% from the mid-term exams and another 10% from participation in lab assignments. In 2013, the eight project assignments had to be solved in pairs instead of individually. The main objectives of this study were to investigate the impact of DPP in authentic learning conditions, and to test the efficiency of two different task distribution strategies.

Seventy-four students (28 females and 46 males) participated in the study; each student was assigned a partner with approximately the same programming background. All participants had previously attended a course in the C programming language in the past, and groups were formed based on the grades earned on that course. The 74 students formed 37 groups, evenly split between the two study conditions to ensure a normal distribution of grades in each condition. Consequently each condition had approximately the same average grade and was composed of both highly skilled and less skilled students. There were 18 groups under the control condition and 19 groups in the experimental condition.

The evaluation procedure was organized as follows. The pair programming methodology and guidelines on the plugin, were presented in a meeting in the third week of the course. In the first half of the semester, a usability test was conducted both to test the consistency of the system and to allow students to become familiar with it and to practice DPP. Students were then asked to fill out a survey; based on their answers some changes were made to the interface, and then the data gathered from the submitted projects was analyzed. To detect groups who had difficulties, asymmetry in group participation levels was examined. Nine groups (five experimental and four

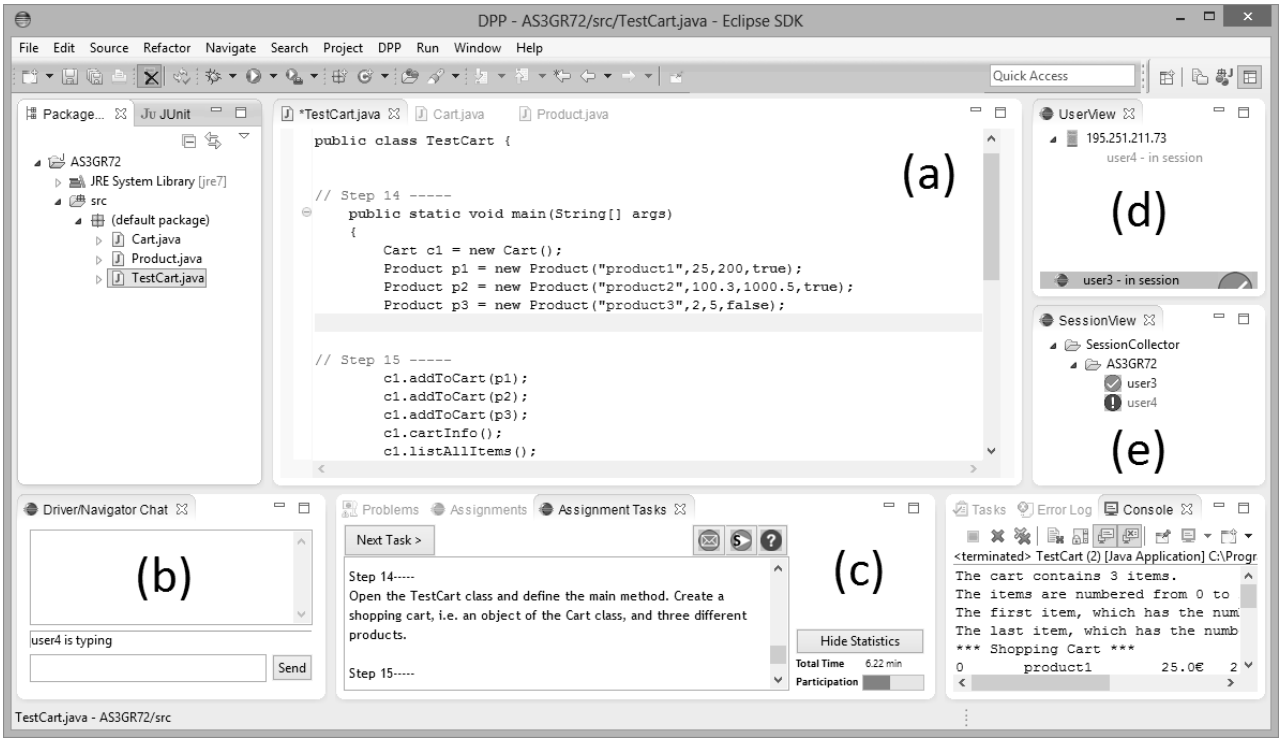

Fig. 1. Screenshot of a DPP session, showing the shared editor (a), the embedded chat tool (b), the script instructions for the task problem statement in Eclipse view (c), awareness indicators of user status (d), and assigned roles (e).

control) had a large difference between the contribution rates of their members; these groups were rearranged with a formation strategy that paired students with similar contribution levels. No changes were made in the other groups because the majority of the students (80%) stated in their responses to the survey that they were satisfied with the chosen partner. Significantly, most of the dissatisfied 20% were from the nine groups that required adjustment. In the second half of the semester the last four projects were carried out, after these rearrangements of groups, and some improvements in the system. Each assignment lasted about eight to ten days. The control condition groups rotated the roles of the "driver" and "navigator" after each task. The experimental condition groups were assigned their roles adaptively, so as to expose each student to diverse activities. In both conditions the system initiated the role switches, but students were also allowed to request a role switch. Throughout the semester periodical tests of logged IP addresses were performed to ensure that the DPP was working properly, namely that students were not co-located. Finally, the last part of the evaluation process consisted of the final examination and an evaluation survey.

V. Evaluation Results

A. Pass Rates

The first item examined was if adopting DPP in a computer science course (traditionally taught by assigning students individual programming projects) had an effect on student pass rates, to answer the research question:

(Q1): Does adopting DPP improve pass rates for the course?

Pass rates for four consecutive years, each with the same instructor, were compared, Table I. The pass rate in 2014 is significantly higher than in previous years, at 69%, indicating that DPP and the group projects had a positive effect on students' learning outcomes. The confidence interval for these rates is between 0.582 and 0.798, meaning that one can say with 95% confidence that about 58% to 80% of students using DPP will pass the course.

A z-test between the pass rates in 2014 and previous years was run to test for a significant difference between the rates, Table I). This was found to be statistically significant at the 95% confidence level (p-value was less than .05), meaning that adopting DPP significantly improved course pass rates (Q1).

TABLE I
Annual Pass Rates

	2011	2012	2013	**2014**
Pass Rate	43%	51%	37%	**69%**
Margin of Error	*0.14*	*0.118*	*0.113*	***0.108***
z-test	z = 2.8, p = .01	z = 2.2, p = .03	z = 3.8, p = .0001	**N/A**

B. Student Performance

Student's final exam grades were analyzed to answer the research question:

(Q2): Does an adaptive distribution of tasks improve student performance more than rotating role switches?

The study examined if the second task distribution mechanism (the experimental condition) did force students to engage with most of the Java concepts and consequently to perform better in exams.

TABLE II
Student performance

	Control Group (n = 31)	Experimental Group (n = 27)	t-test	Multilevel Analysis
Final Examination	M = 4.78 SD = 2.76	M = 5.16 SD = 2.72	t = -0.51, p = .62	-0.29, p = .74
Overall Performance	M = 5.87 SD = 1.89	M = 6.22 SD = 2.28	t = -0.63, p = .53	-0.47 p = .48

Students who had solved only half or fewer of the projects, or who did not take part at the final examination were not considered in the analysis. For the remaining students two different statistical tests were conducted in order to examine if the two study conditions had an effect on their scores. The hierarchical structure of the data required the use of multilevel analysis, but due to the small sample size an independent samples test was also conducted. For CSCL research, it is recommended that results from both analyses be reported to allow a comparison of the two methods [19].

Grades were measured on a scale from 0 to 10, where 10 is "excellent". Although the experimental condition (M = 5.16, SD = 2.72) performed better in the final examination, the tests did not find a statistically significant difference, Table II. As well as final grades, students' overall grades were compared; these were calculated based on final exams, project grades, mid-term exams and laboratory participation. Again, it was not possible to find a statistically significant difference between the control (M = 5.87, SD = 1.89) and experimental conditions (M = 6.22, SD = 2.28).

In view of the small difference in favor of the experimental group, the performance of students who consistently stayed in their assigned roles was examined. In both conditions students were allowed to make individual role switches, and to change or correct code apart from their own. The performance of students who were able to solve the assignments by largely (>80%) remaining in the assigned roles gives an insight into how they would have performed in an ideal situation. Again, the final examination grades and the overall performance were evaluated, Table III. In this case, a Mann-Whitney test was run in addition to multilevel analysis, which revealed that students in the experimental condition had a significant better overall performance (M = 7.7, SD = 1.62) compared to the control condition (M = 5.72, SD = 1.75). A comparison between the final exam grades showed improved performance for the experimental group, almost at a statistically significant level. These results suggest that students who engaged in a wide range of different programming concepts did perform better in the course. This conclusion can also be confirmed by comparing the scores of the experimental condition groups. Students with an even distribution of tasks performed significantly better in the final grade than did the others (t-test results: t(26) = 2.167, p = .04).

Finally, student performance in the last four programming assignments showed that the experimental condition groups performed better on average (M = 8.51) than the control

condition groups (M = 8.19), but a Mann-Whitney test did not reveal a statistically significant difference (U = 93, p = .418). The number of correct assignments, i.e., grades ranging between 8.5 and 10, showed that 60% of the assignments submitted by the control condition groups were correct, compared to 65% for the experimental condition groups.

TABLE III
PERFORMANCE OF STUDENTS WHO REMAINED CONSISTENTLY WITHIN ROLES

	Control Group (n = 14)	Experimental Group (n = 10)	Independent Samples Test	Multilevel Analysis
Final Examination	M = 4.65 SD = 2.6	M = 6.66 SD = 2.19	t = -1.907, p = .07	-2.01, p=.12
Overall Performance	M = 5.72 SD = 1.75	M = 7.7 SD = 1.62	U = 30.0, p = .018	-1.99, p = .036

In answer to the second research question, both strategies appear to be equally effective, except in one special case. The adaptive distribution of tasks proved more effective than rotating role switching when students stayed consistently with the assigned roles. The difference in the results for overall performance was statistically significant, yet the small sample size makes it difficult to generalize the findings.

C. Contribution Rates

The primary goal of combining DPP and collaboration scripts was to address unequal student contributions. Whether or not the methodology achieve the desired outcomes was investigated by answering the research question:

(Q3): Do scripted roles in DPP result in comparable student contributions?

This question was investigated by analyzing the logged information per user level, automatically saved by the system. The system looks at each student's level of participation in the program code, to assess his or her contribution to that code. A limitation of the current design is that it captures only the source code characters written by the driver, and not those written by the navigator (such as highlighting errors in the code or commenting in the chat while the driver writes code). The system does not support content analysis, and this is also hard to achieve, given that students also use external communication channels. The number of exchanged messages and student responses logged revealed that only 19% of the groups used the embedded chat function. Consequently, the contribution rate for each user is calculated as the ratio of the typed code to the total code. The difference between individual contribution rates was calculated within each group, as a figure from 0 to 1, with zero indicating a completely balanced contribution. The average value of this difference was calculated over all the submitted projects; both the control and experimental groups had an average difference of 0.17. Values within the range 0 to 0.2, that is, a participation rate between 40% and 60%, are assumed to be sufficient to indicate symmetrical contributions. To sum up, scripted roles in DPP resulted in comparable student contributions regardless of the task distribution policy used.

D. Distribution of Tasks

Examining the types of tasks solved by each student gives the relative distribution of tasks between team members in the experimental or control conditions. This test answers the research question:

(Q4): Did adaptive role assignments result in a balanced distribution of tasks?

To investigate this question, for each group the number of learning objectives that were not evenly distributed between the group members was counted. The distribution of tasks was then calculated for each individual project as a part of the system's interaction analysis, and the data summarized for all projects. Groups with adaptive role assignments (experimental condition) were found to have a more balanced distribution of learning objectives that did the groups who rotated roles after each task. An independent samples test revealed that this difference was also statistically significant (t-test results: $t(27) = 2.045$, $p = .051$). These results confirm the advantages of adaptive role assignments when compared to rotating roles. Both task distribution policies ensure frequent role switches and an equal number of tasks per student, but the main advantage of the adaptive policy is that students engage with all the learning objectives, which maximizes their learning potential.

In comparing other collaboration-related factors for the two task distribution policies, it was found that all groups completed the assignments in about the same time. They spent 36% (control condition) and 38% (experimental condition) of their time "driving" (i.e., typing program code) and had a comparable frequency of role switches and synchronic program executions. Finally, it was observed that the experimental condition students answered on average more questions on the final examination than did control condition students, but not at a statistically significant level.

E. Group Formation

Based on qualitative and quantitative analysis it can be concluded that the group formation strategy chosen had positive results. Some 84% of the students stated in the final survey that they were satisfied with their partner, and although they had the opportunity to change partners after the midterm exams, they chose to stay with the same one. Additionally, the contribution rates suggest that pairing students with comparable skill levels encourages participation and results in equal student contributions.

The alternative task distribution policy, which was applied in "incompatible" groups (those with a significant difference in contribution rates) also had positive outcomes. Pairing students with comparable contribution levels improved their participation rate in the subsequent assignments. The sample size is too small, however, for this to be a valid result.

F. Group Member Familiarity

Students were asked to rate how well they knew their partner in order to estimate the degree of familiarity in each pair. The responses were based on a scale from 1 (not at all) to 4 (very well), similar to the study in [20]. A significant

correlation was found between the answers of each pair member (r=0.89, p<.01), which strengthens the reliability of the responses. A correlation model was used to investigate the impact of familiarity on individual performance and DPP evaluation. More specifically, the aim was to study if groups with a high level of familiarity did perform better in exams or evaluated the overall DPP experience more positively. Statistical tests found no positive relation between familiarity and performance (r=-0.09, p=.467) or DPP (r=0.184, p=.216). Finally, familiarity ratings were used to investigate if the two study conditions differed at the level of familiarity. The result showed no significant difference, since the control group had an average familiarity score of 2.27 and the experimental group 2.0 (t-test results: t(56) = -1.035, p = .305).

G. Student Feedback

Students were asked to fill out a survey eliciting their opinions of the system and their experience with DPP. The survey consisted of three separate sections that focused on: a) the system's features, and usability issues, b) DPP and collaboration, and c) student comments on problems encountered and suggestions for improvement. The most important findings on system usability are presented in Table IV.

TABLE IV
STUDENT FEEDBACK ON THE USABILITY OF THE SYSTEM (LIKERT SCALE: 1 (STRONGLY DISAGREE) – 5 (STRONGLY AGREE))

	Control Group	Experimental Group	Total
The interface of the system was pleasant to use.	M = 3.68 SD = 0.6	M = 3.62 SD = 0.67	M = 3.65 SD = 0.64
The system was easy to use.	M = 3.64 SD = 0.9	M = 3.66 SD = 0.8	M = 3.65 SD = 0.85
I quickly learned to use the system.	M = 3.79 SD = 0.62	M = 3.93 SD = 0.64	M = 3.86 SD = 0.63
The server response time was acceptable.	M = 2.79 SD = 0.82	M = 2.69 SD = 1.02	M = 2.74 SD = 0.93
Reading the assignment in Eclipse was useful.	M = 4.18 SD = 0.76	M = 4.35 SD = 0.6	M = 4.26 SD = 0.69

Students' comments indicated a positive attitude towards DPP and collaboration. Some 83% of the students stated they would collaborate again in future programming assignments. As an overall experience, DPP was rated with an average score of 3.81 (SD = 0.74) on a scale of 1 (very poor) to 5 (very good). Students confirmed the main benefits of DPP, Table V, and agreed that pair programming made them more responsible in completing the assignments (M = 4.21, SD = 0.72). The only negative experiences were related to the system, with some students reporting server delays and inconsistent editor contents. Technical problems are hard to avoid, but most were fixed during the evaluation phase.

TABLE V
STUDENT FEEDBACK ON DPP AND COLLABORATION (LIKERT SCALE: 1 (STRONGLY DISAGREE) – 5 (STRONGLY AGREE))

With DPP…	Control Group	Experimental Group	Total
Students share knowledge and problem solving skills	M = 3.82 SD = 0.76	M = 3.83 SD = 0.83	M = 3.83 SD = 0.8
Errors in program code can be found sooner	M = 3.65 SD = 0.85	M = 3.93 SD = 0.79	M = 3.79 SD = 0.83
Learning programming is facilitated	M = 4.07 SD = 0.65	M = 3.41 SD = 1.04	M = 3.74 SD = 0.93
Learning to program is more enjoyable	M = 3.93 SD = 0.84	M = 3.66 SD = 0.99	M = 3.79 SD = 0.93
Students can solve more problems on their own	M = 3.72 SD = 0.84	M = 3.45 SD = 1.0	M = 3.58 SD = 0.94
Students are more confident in their assignment solutions	M = 3.64 SD = 0.55	M = 3.59 SD = 1.0	M = 3.61 SD = 0.81

H. Instructor Feedback

The evaluation was carried out throughout the semester in close collaboration with the instructor. The authors provided assistance and technical support whenever requested. As a result they received immediate feedback on the system and were able to improve its functionality whenever feasible. Once the evaluation phase was completed, the instructor filled out a questionnaire giving a detailed report of his experience. The questionnaire asked about the system's usability and the evaluation of DPP, and had some open-ended questions.

Overall, the instructor found the system easy to use and the embedded interaction analysis very useful. He agreed that the adoption of DPP in his course resulted in reduced workload for him, since the number of projects was halved and there were fewer questions. He also found that students' questions were more advanced than usual, and he observed more active participation by the students. The instructor's responses to the open-ended questions were particularly useful because they gave feedback on the use of the system from another point of view. For instance, he noted that the completion time recorded for each assignment indicates its complexity, and possible problems encountered by the students. Furthermore, the recorded "driving" time for each group and its relation to the total time may be used to detect unusual behaviors, such cheating, or performing cooperative instead of collaborative work (cooperative work being work aimed at a common goal, but performed individually, rather than collaboratively). The last part of the evaluation asks the instructor if he would use DPP in the future. He listed the many benefits of DPP for the students, and stated that he would definitely adopt the DPP model in future courses. As a final remark he indicated that an instructor may have to spend an amount of time to create a pool of exercises, form adequate groups and resolve collaboration issues.

VI. DISCUSSION

The results presented above confirm the findings of past PP studies. In [1] pass rates between pairing and non-pairing students were compared, indicating that PP students were more likely to pass the course. Similar results were obtained in studies [2] and [3]. The current findings are consistent with above studies, though students in this study applied DPP and not co-located pair programming.

Research suggests that pairing students with similar skill levels has positive results on motivation and participation [16], [21]. The group formation strategy chosen proved to be a successful approach to forming the groups, since the vast majority of the students were satisfied with their teammate and made equal contributions in the program code.

The rearrangement of some groups at mid-semester was to

separate incompatible students. Since the evaluation took place as a part of a typical Java course, pedagogical reasons made it necessary to change problematic groups. These were not expected to have a significant impact on the final results. In fact, the participation rates of these groups did not affect the overall participation level of each study condition. Before the reassignment, the average differences in contribution rates of control and experimental condition were 0.12 and 0.18 respectively, while after the reassignment the difference was 0.17 in both conditions. Group rearrangements are common in PP classes. For instance, other studies performed periodic pair rotations to separate dysfunctional pairs, and to allow students to collaborate with more of their peers [2], [13], [22].

A comparison between the results of the current study and prior DPP studies indicates similar positive findings on students' perceptions of DPP [8]. For the embedded communication channel, the poor usage of the chat tool confirmed the findings of study [9]. But the assessment of user contributions did not confirm the results reported in [9]. Instead of rare role switches, each session counted several system-driven or user-driven role switches. Compared to the first evaluation of SCEPPSys, in which emerging and scripted roles were evaluated, similar results were observed in the "scripted" condition. Although the mean participation rate of free collaboration was outside the desired limit, students in the scripted condition did achieve a mean difference of 0.165 in their participation rates, identical to those values in the present study.

In the CSCL field, scripted roles are considered more effective than emerging roles [14]. The current study investigated two different approaches to scripted roles: rotating and adaptive role distributions. The latter approach was more effective in terms of performance, but this finding proved statistically significant only for a small number of students. A drawback of the second approach is that these students generally gave lower ratings when asked for feedback on DPP and collaboration in both evaluations of SCEPPSys (though not at a statistically significant level). In light of this finding, after the first evaluation of SCEPPSys the adaptive role switching algorithm was improved. Nevertheless it appears that students still have a lower opinion of the value of DPP. As they stated in the survey, they felt the workload was not evenly distributed when the system performed adaptive role assignments. Probably this impression affected students' willingness to stay on script and points to the need for further improvement of this approach.

Requesting instructor feedback was intended to gain a different perspective on the proposed methodology, and also to compare his experience with those of previous studies. As previously mentioned, the instructor agreed that DPP reduces teachers' workload and that students were able to solve more problems on their own, giving him the opportunity to discuss more advanced issues with them. These findings confirm related results reported in [3], [23] and [24].

Finally, an important finding was that the experimental group students who persisted in their assigned roles performed significantly better than the other students. However, the relatively small sample size means further studies are needed to investigate this issue in more detail.

VII. Conclusion

This paper has presented the results of a long-term evaluation study of the use of DPP in the classroom, and reported findings that contribute to the field of DPP and particularly in evaluating systems for DPP. The reported effects of DPP on student performance and attitude complement the few previous studies in the field.

Moreover, two different approaches to turn taking were studied, and their effects on student performance and the distribution of activities between group members reported. These preliminary results on scripted roles were quite positive and may provide the basis for further research in the CSCL field as well as in DPP.

Future studies should consider the risk of "over-scripting" collaborative activities [15] when designing predefined roles. It is suggested that collaboration scripts should fade out at some point. An adaptive task distribution mechanism can engage students in various types of activities and in the same time allow individual role-taking when symmetry in knowledge building is detected. To this end, future work should include experimentation towards this direction.

References

[1] C. McDowell, B. Hanks, and L. Werner, "Experimenting with pair programming in the classroom," *ACM SIGCSE Bulletin*, vol. 35, no. 3, pp. 60, Sep. 2003.

[2] E. Mendes, L. Al Fakhri, A. Luxton-Reilly, "A Replicated Experiment of Pair-Programming in a 2nd year Software Development and Design Computer Science Course," in *ACM SIGCSE Bulletin*, vol. 38, no. 3, pp. 108-112, 2006.

[3] N. Nagappan, L. Williams. M. Ferzli, E. Wiebe, K. Yang, C. Miller, S. Balik, "Improving the CS1 Experience with Pair Programming," *ACM SIGCSE Bulletin*, vol. 35, no. 1, pp. 359 – 362, 2003.

[4] B. Hanks, "Empirical evaluation of distributed pair programming," *International Journal of Human-Computer Studies*, vol. 66, no. 7, pp. 530–544, Jul. 2008.

[5] D. Stotts, L. Williams, N. Nagappan, P. Baheti, D. Jen, and A. Jackson, "Virtual Teaming: Experiments and Experiences with Distributed Pair Programming," In *Extreme Programming and Agile Methods-XP/Agile Universe*, pp. 129-141, 2003.

[6] D. Tsompanoudi, and M. Satratzemi, "Enhancing Adaptivity and Intelligent Tutoring in Distributed Pair Programming Systems to Support Novice Programmers," in *Proceedings of CSEDU'2011*, pp. 339-344, 2011.

[7] C. Ho, S. Raha, E. Gehringer, and L. Williams, "Sangam – A Distributed Pair Programming Plug-in for Eclipse", In *Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange*, pp. 73–77, 2004.

[8] K. E. Boyer, A. A. Dwight, R. T. Fondren, M. A. Vouk, and J. C. Lester, "A development environment for distributed synchronous collaborative programming," *ACM SIGCSE Bulletin*, vol. 40, no. 3, pp. 158, 2008.

[9] T. Schümmer and S. Lukosch, "Understanding Tools and Practices for Distributed Pair Programming," *Journal of Universal Computer Science*, vol. 15, no. 16, pp. 3101-3125, 2009.

[10] D. Tsompanoudi, M. Satratzemi and S. Xinogalos, "Exploring the Effects of Collaboration Scripts Embedded in a Distributed Pair Programming System," in *Proceedings of ITiCSE'2013*, pp. 225-230, 2013.

[11] D. Tsompanoudi and M. Satratzemi, "A web-based Authoring Tool for Scripting Distributed Pair Programming," in *Proceedings of ICALT'2014*, pp. 259-263, 2014.

[12] L. Kobbe, A. Weinberger, P. Dillenbourg, A. Harrer, R. Hämäläinen, P. Häkkinen, and F. Fischer, "Specifying computer-supported collaboration scripts," *ijCSCL*, vol. 2, no. 2–3, pp. 211–224, Sep. 2007.
[13] L. Williams, "Lessons learned from seven years of pair programming at North Carolina State University," in *ACM SIGCSE Bulletin*, vol. 39, no. 4, pp. 79-83, Dec. 2007.
[14] J. W. Strijbos and A. Weinberger, "Emerging and scripted roles in computer-supported collaborative learning," *Computers in Human Behavior*, vol. 26, no. 4, pp. 491–494, Jul. 2010.
[15] P. Dillenbourg, "Over-scripting CSCL: The risks of blending collaborative learning with instructional design," *Three worlds of CSCL. Can we support CSCL?*, pp. 61-91, 2002.
[16] L. Williams, L. Layman, J. Osborne and N. Katira, "Examining the Compatibility of Student Pair Programmers," *Agile Conference*, 2006.
[17] G. Braught, J. Maccormick, and T. Wahls, "The Benefits of Pairing by Ability", In *Proceedings of the 41st ACM technical symposium on Computer science education*, pp. 249–253, 2010.
[18] L. J. Barker, "When do Group Projects Widen the Student Experience Gap?,", In *Proceedings of ITiCSE '05*, pp. 276–280, 2005.
[19] U. Cress, "The need for considering multilevel analysis in CSCL research—An appeal for the use of more advanced statistical methods," *International Journal of Computer-Supported Collaborative Learning*, vol. 3, no. 1, pp. 69-84, 2008.
[20] J. Janssen, G. Erkens, P. A. Kirschner and G. Kanselaar. "Influence of group member familiarity on online collaborative learning," *Computers in Human Behavior*, vol. 25, no. 1, pp. 161-170, 2009.
[21] N. Z. Zacharis, "Measuring the Effects of Virtual Pair Programming in an Introductory Programming Java Course," *IEEE Trans. Education*, vol. 54, no. 1, pp. 168-170, Feb. 2011.
[22] G. Braught, T. Wahls and L. M. Eby, "The case for pair programming in the computer science classroom." *ACM Transactions on Computing Education (TOCE)*, vol. 11, no. 1, 2011.
[23] L. Williams, K. Yang, E. Wiebe, M. Ferzli and C. Miller, "Pair Programming in an Introductory Computer Science Course: Initial Results and Recommendations," In *OOPSLA Educator's Symposium*, 2002.
[24] B. Hanks, "Problems Encounte-red by Novice Pair Programmers," In *International Computing Education Research Workshop*, pp. 159-164, 2007.

Despina Tsompanoudi is a Ph.D. candidate in the Department of Applied Informatics at the University of Macedonia, Greece. She holds a B.Sc. degree in informatics (2003) and a M.Sc. degree in information systems (2006).

Maya Satratzemi is a professor in the Department of Applied Informatics. Her current main research interests lie in the area of educational programming environments and techniques, didactics of informatics, 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).

Stelios Xinogalos is an assistant professor at the Department of Applied Informatics. He is author or co-author of more than 70 research papers on programming environments and techniques, object-oriented design and programming, educational environments and games for programming.