Pair Programming for Software Engineering Education: An Empirical Study

Kavitha Karthiekheyan[1], Irfan Ahmed[2], and Jalaja Jayalakshmi[1]
[1]Department of Computer Applications, Kumaraguru College of Technology, India
[2]Department of Computer Applications, Sri Krishna College of Engineering and Technology, India

Abstract: *As an iterative and incremental methodology, agile software has helped a lot in evolving solutions from self-organizing, cross-functional teams. Pair programming is a type of agile software development technique where two programmers work together with one computer for developing the required software. This paper reports the results of a pair programming exercise carried out with a set of one hundred and twelve post graduate students, who developed small applications as a part of their software development laboratory course at Kumaraguru College of Technology (KCT) during the academic year 2012-2013 and 2013-2014. The objective of this research is to investigate the effect of adopting pair programming as a pedagogical tool in Indian higher educational setting. Purposeful pair programming modules were deployed in various phases of software development and the results revealed that pair programming is not only an useful approach in teaching computer programming but also facilitate effective knowledge sharing among students. Further, the effectiveness of pair programming was realized to a greater extent during the designing and coding phases of software development. Practicing pair programming also enables the students to develop their collaborative skills, which is crucial to an industrial working environment.*

Keywords: *Agile software development, collaborative learning, knowledge sharing, pair programming, software engineering, education.*

Received December 13, 2014; accepted April 26, 2015

1. Introduction

Agile software development contains immense potential for delivering increased productivity, quality and project success rate in software development projects by synergizing considerable tacit knowledge. In addition to promoting proactive planning, it also encourages rapid and flexible response to change by providing collaborative environment. Extreme Programming [3] is one of the most widely recognized agile software development methods, which focuses on disseminating knowledge through collaborative practices such as pair programming, planning game and retrospectives. Pair Programming is an extreme programming practice, where two programmers mutually collaborate at the same workstation to acquire added up knowledge and experience on everyday basis. It works around the principle of sharing work and expertise across teams.

Organizations focusing on developing software practice pair programming where the programmers collaborate as pairs by sharing a single computer working with the same design, algorithm, code, or test etc., [24]. While one member of the pair, namely the driver types at the computer or writes down a design, the other who assumes the role of the navigator, observes the work of the driver to ensure objectivity, logic and process flexibility [6]. Research results have provided evidence for the efficacy of pair programming [2]. Empirical evidence suggests that two programmers working in collaboration can be twice more effective in terms of speed and the possibility of finding solution when compared to two programmers working individually. The effectiveness of the pair programming stems from effective and frequent brainstorming among the pairs [7]. In addition, it also facilitates frequent exchange of roles between pairs. Further, through constant code reviewing, pair programming also minimizes defects to a significant extent, thus resulting in the development of an error-free quality product [21].

Pair Programming (PP) has been offering considerable promise as a strategy to learn programming in academic environments. Engaging in collaborative activities such as pair programming engenders greater participation and better interaction among learners when compared to programming done individually. Studies revealed that pair programming complimented the learning process commendably within a short period of time [24]. Further, it also helped students to gain real time practical experience of software development through knowledge sharing and collaboration [20]. While programming in pairs, both partners tend to discuss and work on the given problem, thus sharing their experience and knowledge [1, 17, 23].

In this study, the researcher explores the various aspects of pair programming which was used as a teaching-learning methodology in a software development laboratory course as a part of Master of

Computer Applications program. Further, this study also investigates the impact of pair programming practices on software development by studying the effectiveness of pair programming

a) In various phases of software development.
b) In facilitating collaborative learning through knowledge sharing. The analysis was based on data sources gathered for about two semesters in a post graduate computer applications course.

The outline of the paper is as follows. The review of literature is presented in section 2. The work proposed is discussed in section 3 and the analysis and interpretation of the collected data is presented in Section 4. Student's perception on pair programming is discussed in Sections 5 and 6 discusses the conclusion part of the work.

2. Existing Studies on Pair Programming

A comprehensive review of literature was done in order to understand the impact of pair programming exercise on teaching learning process. The ability to work as part of a cross-disciplinary team in industry has been highlighted by Scott and Wilson [19]. Kuppusami and Vivekanandan [10] experimented with computer science course students comparing the learning efficiency of students who adopted pair programming with those using traditional method for laboratory exercises for a short duration.

The cost-effectiveness of PP and the potential contained in the same for developing codes with a few errors have been demonstrated by Muller [12]. According to Lui *et al*. [11], pair programming promotes not only quality programming skills, but also enhances responsibility, mentoring, teamwork in addition to providing an increased sense of enjoyment. Vanhanen and Korpi [21] demonstrated their experiences of using PP extensively in an industrial project.

An extensive and substantial case study on pair programming was carried out in software development courses at the University of Dortmund, Germany by Bipp *et al*. [5]. Thirteen software development teams with a total of 100 students took part in the experiments. The groups were as follows: In one set, the group members worked on their projects in pairs. Not only did these teams produce nearly the same number of codes as the teams of individual workers in the same period, but their codes were easier to read and understand thus facilitating easy detection and correction of errors. Research conducted by Begel and Simon [4] also brought to fore the fact that freshly inducted software developers often struggled to adequately communicate, when they were in need of assistance or while they were struggling with a problem. Pikkarainen *et al*. [14] studied the communication aspect of agile software development and concluded that agile practices improve both formal and informal communication among team members.

On reviewing 66 studies, Salleh *at el*. [18] identified certain psychosocial factors such as compatibility, personality and gender issues, which affect the effectiveness of pair programming among students. The effects of pair programming on knowledge transfer and the resulting sense of fulfillment experienced by students were reported by Venkatesan and Sankar [22]. Hannay *et al*. [8] observed that the personality of the pairs engaged in pair programming could be a valid predictor for long-term team performance. Salleh *at el*. [18] presented evidence related to the effectiveness of Pair Programming (PP) as a pedagogical tool in higher education CS/SE courses. Naufal and Hui [13] conducted a study to investigate the influence of learning style preference on learning C++ language, especially in terms of the quality of codes written and the number of errors that occurred in programming. Radermacher and Walia [16] attempted to specifically identify the presence of knowledge deficiencies among the students to influence curriculum changes in order to address the same.

Studies reported in literature mostly involved experiments conducted for a limited duration ranging from a few laboratory sessions to a few months. Further, only a few studies in the Indian educational context have been reported so far. The current study aims to plug the gap by undertaking a controlled experiment and extending it to a longer duration i.e. a period of six-months for each batch of students for the same laboratory course. In addition to exploring the knowledge sharing aspect of PP, it also records the effectiveness of PP during various phases of software development, as perceived by students.

3. Proposed Work

The strength of knowledge sharing through pair programming was felt when the students of the Master of Computer Applications (MCA) program struggled a lot initially while developing applications, owing to their different educational backgrounds [9]. Based on the researcher's experiences and insights drawn from existing literature, the researcher proposed to study the effects and experiences of the pair programming concept. Initially, the factors including the usefulness of pair programming, reduced errors, collaborative skills, proactive learning and knowledge improvement were taken into consideration. The objectives of the study are:

- To test the association between the overall scores secured by students in similar software development projects, irrespective of whether they worked in pairs or as individuals.

- To test the significant difference in the phase-wise scores secured by students in similar software development projects, irrespective of whether they worked in pairs or as individuals during the various phases of software development.
- To study students perceptions on the effectiveness of pair programming during various phases of software development.

3.1. Experimental Methodology and Context

In order to facilitate learning process of students in the computer applications course, the study investigated the use of pair programming as a teaching methodology and investigated its effectiveness on students overall learning process.

Formal lists of questions were prepared and the responses were analyzed using standard statistical techniques. Two questionnaires with close-ended questions containing a 5-point rating scale were designed. The students were made to fill an entry questionnaire consisting of ten questions to assess their level of exposure to programming tasks, partner preferences etc. The worksheet also contained twelve open-ended questions, which allow the students to provide their own answers in an unprompted manner, thus yielding qualitative data. After the completion of the project, an exit questionnaire containing twenty questions on knowledge sharing, tool learning, pair programming effectiveness during various phases of software development and general experiences on pair programming Table 1 was administered to each student practising PP. Also, unstructured interviews were conducted to understand their pair programming experiences and clarify their responses to questionnaires.

The Unified Process (UP) is an iterative software development process framework [15]. The UP determines a project life cycle as consisting of four phases namely inception, elaboration, construction and transition.

In the inception phase, the business case which includes business context, success factors and financial forecast is established. The primary objective of the elaboration phase is to mitigate the key risk items identified by analysis till the end of this phase. The objective of the construction phase is to build the software system. The bulk of the coding takes place in this phase and the first external release of the software also happen at this stage. Finally, the transition phase helps to 'transit' the system from development into production, thus enabling the end user to understand the system.

The pair programming experiment was carried out for two batches (2012-13 and 2013-14) in software development laboratory course in the fifth semester of the MCA Program. The credit for the course weighted 50% out of the total credits among the other laboratory courses carried out in that semester. The curriculum doesn’t contain an associated theory component. The students were already exposed to subjects like software engineering, databases and object oriented analysis and design techniques, which are essential for developing software applications during laboratory sessions. The students were expected to use IBM rational suite and microsoft visual studio software. While developing the application, the students were instructed to follow the Unified Process model for developing the software as Rational Suite recommends it. Since they lacked prior knowledge of the above-mentioned tools, they were asked to explore the various functionalities and to develop small applications on their own.

To begin with, a batch of fifty-five third year MCA students participated in the study. The study was carried out in a controlled experimental setup. While pair and solo programming were considered as independent variables, improved knowledge sharing and work quality were treated as dependent variables. Eighteen pairs were formed by Pair Programming Information System (PPIS) based on student responses to the following factors such as willingness to participate in the experiment, partner preferences, level of knowledge, cumulative grade point average secured till the previous semester and their level of expertise in developing software applications and tool usage. The remaining 19 students were made to work individually. The students who belonged to this group either preferred working solo or were capable of working on their own.

Most students preferred to work with the same gender and had no problems working with partner of any knowledge level. The major intent of the study is to enable the average and slow learners to learn and display improved performance in the laboratory course. Hence, the students were categorized into 4 levels based on the cumulative grade point average secured. The students were grouped as follows: Level 1 consisting of top performers, level 2 the above average performers, level 3 the average category and level 4 the slow learners. Students who were in level 4 and 3 were either paired with students in level 1 or level 2 in order to facilitate effective knowledge sharing. The students who were asked to work in solo either belonged to level 1 or 2, considering the fact that they were capable of learning and working on their own. Yet another batch of fifty seven third year MCA students was also involved in the study. All students were made to work in pairs with the intent of collecting only their pair programming experiences. No comparisons were drawn between the project outcomes of these batches. This data was used to correlate the variables Table 3.

They were given the freedom of choosing their pairs in order to know their level of satisfaction while working with a partner of their choice.

3.2. The Controlled Experiment

A process framework was designed in order to carry out the pair programming exercise systematically Figure 1. Appropriate user interfaces available in the framework enabled student respondents and the assessors to record data easily. Once the students were found to acquire the requisite understanding about pair programming, they were allowed to access online software PPIS, which forms a part of the framework. PPIS enabled the students to fill the entry and exit questionnaire online. The questionnaire entries were stored in appropriate databases [9].

For the first batch, eighteen software application development projects with equal levels of difficulty were chosen for the experiment by the faculty. These projects were randomly allotted to the students and the scope and requirements were clearly explained to contextualize the results. These tasks were executed during separate lab sessions of five hours duration per week. The same project title was allocated to pairs and solo programmers in order to enable easy comparison of outcomes and the milestones for the projects were also announced. For the second batch, twenty eight similar application development projects were given to student pairs with the main intention of collecting their general experiences related to pair programming. One student volunteered to work on his own. The pairs were asked to interchange driver-navigator roles once in the middle of each laboratory session to ensure equal contribution to the project.

During the lab experiment, the students were asked to record their experiences individually for each lab session. In order to extract and record the software development and learning experiences of those students working in pairs, they were made to fill in a worksheet as detailed in Table 1. Subsequently, details related to knowledge sharing and transfers were also collected. After the tasks were completed, the students were asked to fill in an online exit questionnaire, specifically designed to collect their views on pair programming, knowledge sharing, tool learning and collaborative skill development.

Each application developed by the student was assessed phase-wise and their comments were recorded in the software. Artifacts like requirements management plan, vision document, use case documents, Unified Modeling Language (UML) diagrams, code developed for each module, test cases, test plans and test results were also taken into consideration for assessment purposes. Further, each student was also asked to demonstrate the application developed by them and the data thus collected was analyzed using appropriate tools. The results of the analysis are presented in the following sections.

The previous studies reported in literature have not used a complete process framework that is fully automated. When the entry questionnaire is filled by the respondent online, PPIS would automatically suggest pairs based on student preferences. It would also suggest pairs randomly on demand. Once the data entry is complete for the questionnaires, worksheets and assessment sheets, the data will be stored in a database that can be exported in excel format, which in turn can be fed into the analysis tools.

As a means of achieving validity, during the pre experiment phase, the assessors gave a presentation on pair programming, covering details related to the type of problem that would be given to the students during the experiment and the desired outcomes in the laboratory sessions. Following this, the basic features of the IBM rational suite and microsoft visual studio environment were demonstrated by the faculty.

Clear, consistent and unambiguous instructions regarding criteria and method of assessment were also given. Therefore, even before starting the experiment, the student respondents were clear about the tasks to be carried out during the lab sessions, phase-wise artifacts that would be assessed and also about the various assessment methods such as awarding grades for the phase-wise artifacts, presentations and viva. The students were given sufficient time to explore the concepts of the prescribed software by referring to e-books and lab manuals. Care was taken to ensure that the pairs interchanged among themselves and shifted roles frequently to enable knowledge transfer and to ensure equal contribution to the work. To ensure accurate recording of data, all the laboratory sessions were coordinated and closely monitored by two faculty members.

Both the assessors involved in the study have more than ten years of teaching experience and have prior experience in handling software development related subjects. They have also guided more than fifty student projects and have rich expertise in using rational software. The assessors conducted the lab sessions methodically and used a standardized scheme Table 2 for assessment purposes. They followed clear and systematic recording procedures in order to ensure valid and reliable data. Each student was assessed individually by both the assessors and their scores obtained were more or less the same with a maximum difference of five percent, thus ensuring inter-rater reliability.

Pair programming experiment was carried out for different batches of students in various laboratories. Student respondents who underwent pair programming exercise in the software development lab had also worked in pairs in visual programming laboratory course conducted in the previous semester. The students were assessed by the same two assessors and under the same standards and guidelines. The results seem to be promising [1] and students had opined that pair programming was very useful to them and helped them in sharing knowledge and learning a tool with a good interface. The students had also scored well in final semester exams in both the laboratories, with a further improvement of scores in levels 3 and 4. The results obtained from the experiments conducted in both the labs seem to be consistent, thus increasing the reliability of the data collected.

Generally, the attitudes and behavior of student respondents might not be consistent. At times, it is possible that the questions may not be interpreted by them as they are intended to be. In all likelihood, the student respondents may rate a factor without understanding the question carefully, thus yielding imprecise data and creating a threat to the validity of the data. This problem was addressed by designing questions that can be both clearly and easily understood by student respondents.

4. Analysis and Interpretation of Data

The collected data was analyzed using IBM SPSS predictive analytics software.

4.1. Study of the Correlation Between Variables

The data for deriving this table was taken from the exit questionnaire collected from both batches of student respondents. The variables used in the table are explained as follows:

1. Pair support-support and coordination rendered by the pair during pair programming session.
2. Levels of awareness-during pair programming, the partner constantly watch the activities of his/her pair, thus increasing the awareness levels.
3. Reduced Errors-the extent to which constant code reviewing during PP helped to reduce errors.
4. Collaborative skills- improvement in collaborative skills after pair programming.
5. Construction phase activities-effectiveness of PP during the construction phase of software development.
6. Proactive Learning-Improvement of work quality and skills through effective learning.

The ratings given by the respondents for various factors about pair programming were stored in an excel sheet and this data was used to find the correlation between the variables. Factor analysis and pearson correlation tests were used to find the correlation and the results are exhibited in Table 3. It can be seen that the highest correlation of 0.598 was observed for the following two variables namely pair support and reduced errors. The correlation value was found to be significant. During pair programming, since one of the pair, namely the navigator looks for errors and bugs made by the driver and correct it then and there, the errors while developing software were minimized to a significant extent.

Table 1. Sample questions asked in the questionnaire.

Entry Questionnaire	Exit Questionnaire	Worksheet
Rate your level of understanding on the subjects Software engineering and object oriented analysis and design.	Do you think working in pairs was useful?	Lines of code developed
Mention the number of software applications developed so far	Do you see yourself getting better in developing collaborative skills?	Types of errors and time spent for debugging
Rate your level of familiarity of the concept 'Pair Programming'.	To what extent having a second opinion helped you to avoid common mistakes, reduced errors thus saving time?	Contribution of partner in correcting errors
Mention the preferred level of your partner while doing pair programming.	Did pair programming improve your work quality and skills?	Pair programming experience in the session
How far you will be comfortable working with a different gender?	Effectiveness of pair programming in various phases.	Difficulties faced in the session if any

Table 2. Artifacts used for assessment.

Inception phase artifacts (10 marks)	**Elaboration phase artifacts (10 marks)**	**Construction phase artifacts (10 marks)**	**Transition phase artifacts (10 marks)**	**Project execution (10 marks)**
Requirement management plan Vision document Use case ocument	Class diagram Use case diagram Sequence Diagram Activity diagram	Lines of Code Readability of code Complexity of code Understandability of code Defect density	Test cases developed Test plan Test cases passed Test results	Correctness and effectiveness of output Alternative solution/ Thinking different Viva Project demo

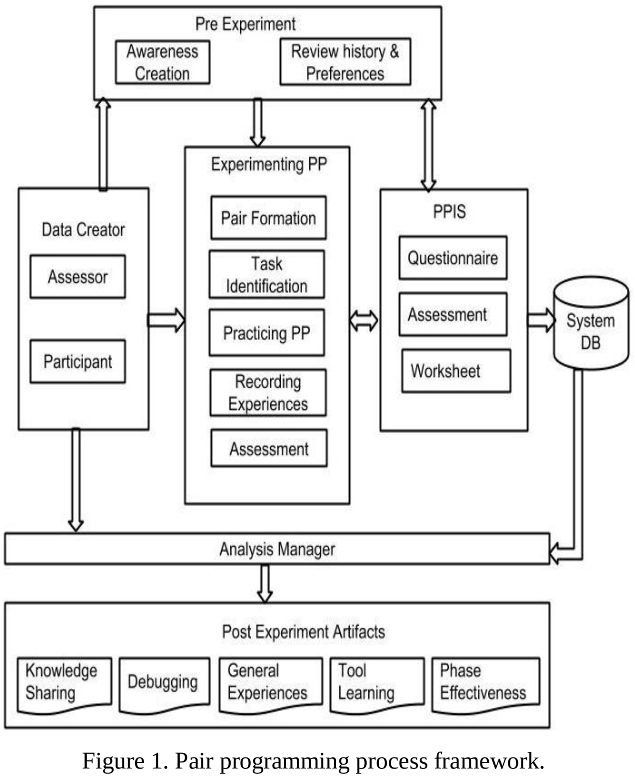

Figure 1. Pair programming process framework.

Table 3. Correlation between vsariables.

	Pair Support	Levels of awareness	Collaborative Skills	Reduced Errors	Construction Phase activities	Proactive Learning
Pair Support	1	.444**	.331**	.598**	.208*	.002
Levels of awareness		1	.591**	.459**	.314**	-.120
Collaborative Skills			1	.538**	.360**	.010
Reduced Errors				1	.317**	-.164
Construction Phase activities					1	.565**
Proactive Learning						1

** Correlation is significant at 0.01 level

The next highest correlation was found for the variable levels of awareness and the collaborative skills, with a correlation value of 0.591. Students normally seemed to prefer collaborating among themselves to obtain new ideas and also to seek advice from one another. Thus, those students who learn collaboratively and benefit from the ideas and opinions of others seem to have an increased level of awareness. Further, there also seemed to be a good correlation between the variable collaborative skills with other variables namely reduced errors, construction phase activities and proactive learning. The correlation between the variable proactive learning and the variables pair support, levels of awareness, collaborative skills and reduced errors was also found to be very low and insignificant. The only variable that correlates with proactive learning seems to be the construction phase activity. Thus, it can be concluded that the construction phase of software development provides more opportunities and contains more scope for proactive learning.

4.2. Association Between Overall Scores Secured by Students Undergoing Pair and Solo Programming

A paired t test was carried out to test whether there is any difference between the outcomes of similar projects that were carried out in pairs or solo. The hypothesis formulated for the same is as follows:

- H_0: There is no significant difference in the overall scores secured by students in similar software development projects, irrespective of whether they worked in pairs or as individuals.

The outcomes of the project done using pair and solo programming techniques were evaluated by the faculty. A phase-wise evaluation was carried out considering the artifacts produced in each phase of the software development. The overall scores comprised of the evaluation of the student's projects in various phases namely the inception, elaboration, construction and the transition phase, with a maximum score of 10 for each phase and 10 for the project execution, aggregating to a maximum overall score of 50. A paired t-test was used to compare the overall scores of pair and solo programmers and the results are shown in Table 4.

Table 4. Overall scores of pair/solo programmers.

		Mean	SD	t	df	Sig. (2-tailed)
Overall Scores for projects secured by Pairs/Individuals	Pair	39.16	3.43	4.469	17	.000
	Solo	32.22	5.44			

From the results, it can be concluded that the significance value for the paired t test is less than 0.05, and is highly significant. Therefore, the null hypothesis can be rejected and the alternative hypothesis that there is difference in the scores of the pair programmers and solo programmers on similar projects stands accepted. It can also be seen that the mean of the overall scores for projects was observed to be high in the case of pair development when compared to scores for projects developed by individuals Figure 2. The learners seem to engage in more creative software development when compared to other areas of development. Further, since pairs generated better ideas than solo programmers, their overall scores stand high.

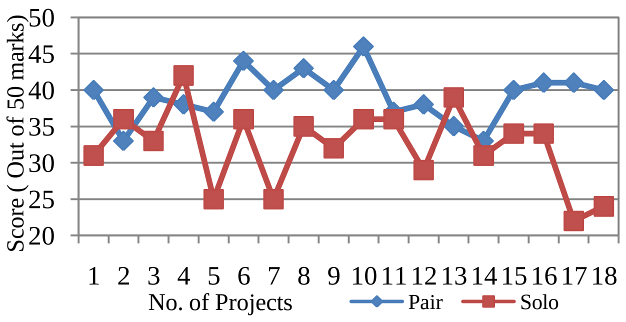

Figure 2. Pair-Solo score comparison.

From Figure 2, it can be understood that the highest score of 46 was secured by the students who worked as pairs, belonging to levels 1 and 3. The pair seems to have shared knowledge and brainstormed on various issues, thus emerging with a better score. The lowest score of 33 was secured by a student pair who belonged to level 3 and level 2. As per the worksheet data collected from the pair, it can be understood that the pair collaboration and knowledge sharing was low, resulting in low scores among students. Similarly, while looking at the scores secured by solo programmers, the highest score 42 was secured by a level 1 student and the lowest score 22, by a level 2 student. The student who scored low among the solo programmers admitted that he was not so efficient in software design and also that he spent more time on debugging. The student further opined that his score may have improved if he had undergone pair programming. Thus, it can be inferred that students working as pairs and developing software projects outperformed those working solo.

Table 5. Phase-Wise Scores of Pair/Solo Programmers.

Development Phase	Programming style	Mean	Std. Deviation	t	df	Sig. (2-tailed)
Inception	Pair	7.166	1.24853	1.824	17	.086
	Solo	6.166	2.38253			
Elaboration	Pair	8.388	1.14475	2.803	17	.012
	Solo	7.111	1.49071			
Construction	Pair	8.000	1.02899	3.051	17	.007
	Solo	6.777	1.47750			
Transition	Pair	7.555	.98352	5.208	17	.000
	Solo	4.833	2.40710			

4.3. Association Between Phase-Wise Scores Secured by Students Undergoing Pair and Solo Programming

A paired t test was carried out to test whether there is any difference between the outcomes of similar projects that were carried out in pairs or solo during different phases of software development. The hypothesis formulated for the same is as follows:

- H_0: There is no significant difference in scores secured by students in similar software development projects, irrespective of the fact whether they worked in pairs or as individuals during various phases.

A phase-wise evaluation was carried out considering the artifacts produced during each phase of software development and the scores were recorded in the assessment sheet by the assessors. In the inception phase, artifacts such as vision document, requirements management plan and use case documents were considered. In the elaboration phase, diagrams such as activity, use case, sequence and class were considered. The construction phase was evaluated by analyzing the code developed by the students, considering factors such as code quality, readability, defect density and productivity in terms of lines of code. Finally, the evaluation criteria for transition phase consisted of test cases, test plans prepared and the test results produced by automated testing tools. The comparisons of the phase-wise scores are shown in Table 5. It can be observed that the scores obtained through pair programming are high in all the four phases of software development. It can be concluded from the results that the significance value for the paired t test was less than 0.05 and highly significant for H_0 b, H_0 c and H_0 d. Therefore, the null hypothesis stands rejected and the alternative hypothesis stands supported. During these three phases, the pairs collaborated effectively, generating better ideas and showing better productivity, when compared to solo programmers. From the scores obtained in the inception phase, it can be observed that the significance value for the paired t test is greater than 0.05, thus validating the null hypothesis H_0 a. Thus, it can be understood that the students can work effectively even as individuals during the inception phase, where the requirements are gathered and planning is completed. It can be inferred that even though the pair score in the transition phase is high, the students have expressed that pair programming in this phase is less effective when compared to other phases of software development.

5. Students Perception of the Effectiveness of Pair Programming in Various Phases of Software Development

Table 6. Effectiveness of pair programming in various phases of software development.

Phases	Minimum	Maximum	Mean	Std. Deviation
Inception	2.00	5.00	3.3370	.90514
Elaboration	3.00	5.00	4.2935	.62085
Construction	3.00	5.00	3.7717	.68103
Transition	1.00	4.00	2.4457	.78955

A questionnaire was used to rate the effectiveness of pair programming in the various software development phases. From Table 6, it can be inferred that the mean score of students is higher in the elaboration phase, which involves the design activities of the project. The prospects of achieving a better design by collaborating and sharing ideas also seem brighter. While working in pairs, students think of alternative designs and problems can be corrected immediately through brainstorming. The next phase in which students felt pair programming to be effective is the construction phase, which involves coding. Since one of the pair acts as the navigator, he/she looks into the code typed by the navigator, finds errors and tries to correct it immediately. This minimizes errors to a considerable extent. Students rated that pair programming is not so effective in the transition phase, which involves testing activities Figure 3. The students easily tested the developed application by referring to the test plans and test cases using automated testing tools. Students felt that testing activity is manageable even while working individually.

5.1. Students Performance in Final Exam

The student respondents who participated in these experiments were selected upon base criteria namely their Cumulative Grade Point Average (CGPA) scores obtained during their previous semesters of the course and were categorized into four levels. The students were continuously monitored and assessed through two internal tests and one end semester examination with a viva component.

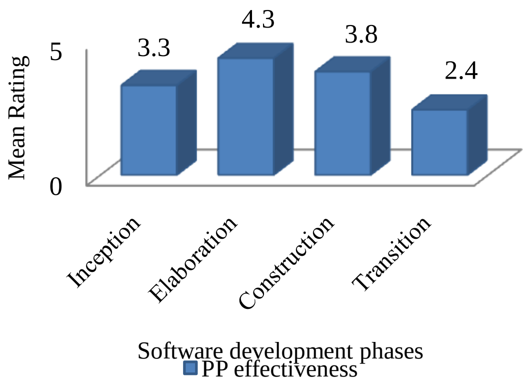

Figure 3. Pair programming effectiveness.

The software development exercise given for end semester examination included problems with higher levels of difficulty when compared to those that they would normally solve in regular laboratory sessions. Each student respondent was assigned with one individual problem. Finally the assessors evaluated the outcome for 100 marks for program output and viva respectively. The criteria for evaluating program output encompassed the correctness of phase-wise artifacts, requisite output and problem understanding. The evaluation was also complimented by a viva session that contributed to 20% of the total marks. This was done to test the efficiency of the student respondents to explain about the tool features, requirement analysis and design which they have used. The test results revealed that the student respondents had gained a significant knowledge about tool usage and programming tactics. In addition, it was also noted that levels 3 and 4 students demonstrated better performance, scoring in the range of 71-90. The results of another laboratory course which adopted solo programming conducted during the same period were compared to benchmark the performance outcomes of those students who underwent pair programming, Figure 4. The results indicate clearly that pair programming experiments considerably enhances the students performance. To substantiate, it was recorded that even level 3 and level 4 students felt that pair programming sessions boosted their performance. Students who belonged to levels 1 and 2 performed well as they do in other laboratories which insist on solo programming. Also, these students expressed that they enjoyed pair programming and also that they gained a sense of satisfaction in sharing their knowledge with peers. A few students who were involved in solo programming also expressed their willingness to adopt pair programming in future.

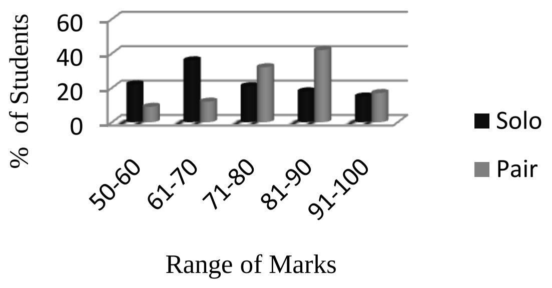

Figure 4. Student's performance - solo and pair programming.

6. Conclusions and Scope for Further Work

Collaborative work is now being looked upon more seriously than ever in teaching-learning process. This was the impetus for carrying out the above-reported research. The study reports the results of preliminary work carried out in implementing pair programming as a teaching methodology. The results of the study conducted in the context of a programming laboratory course appear to be positive and also reveal the potential of PP in improving both programming practice and collaborative skills. These results propelled the researcher in experimenting with the benefits of collaborative skills on a large scale. Following this, the researcher developed a process framework for pair programming and experimented the effects of the same for a longer duration. Most student respondents have acknowledged in the questionnaire that practicing in pairs did help them experience a sense of reward and accomplishment. The students seem to believe that this experience would prepare them for their transit into an industrial setting which is open to them immediately after completing the course. Thus pair programming can influence practitioners to use it as a teaching learning methodology in laboratories, where more creativity is expected and where the students are required to learn on their own, with minimal support from the faculty. However, a few students who worked in paired teams expressed their dissatisfaction with team work due to lack of cooperation of the pair. While looking at the software project scores of the students who underwent solo and pair programming, it can be concluded that the scores of pair programmers are higher than those of the solo programmers. These results seem to be consistent with the results of previous studies reported in literature. As an extension of the present work, the researchers are currently working on developing software tools that would help in the evaluation and analysis of the experimental sessions. The study can be further expanded by choosing student groups with 4/6 students with mixed knowledge levels, within which the pairs can be formed. The pairs can be interchanged after a specified duration within the group, thus enabling easy knowledge transfer among the members of the group.

Acknowledgements

The authors would like to thank the Management of Kumaraguru College of Technology, Coimbatore, India for providing the necessary support to conduct the experiment and collect the necessary data for research. We would like to thank all the students who participated in the study and faculty friends who supported the study.

References

[1] Akerkar R. and Sajja P., *Knowledge-Based Systems*, Jones and Bartlett Publishers, 2010.

[2] Arisholm E., Gallis H., Dybå T., and Sjoberg D., "Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise," *IEEE Transactions on Software Engineering*, vol. 33, no. 2, pp. 65-86, 2007.

[3] Beck K., *Extreme Programming Explained*, Addison-Wesley Langman Publishing, 2000.

[4] Begel A. and Simon B., "Novice Software Developers, All over Again," *in Proceedings of the 4th International Workshop on Computing Education Research*, Sydney, pp. 3-14, 2008.

[5] Bipp T., Lepper A., and Schmedding D., "Pair Programming in Software Development Teams-An Empirical Study of its Benefits," *Information and Software Technology*, vol. 50, no. 3, pp. 231-240, 2008.

[6] Cockburn A. and Williams L., *The Costs and Benefits of Pair Programming*, Extreme Programming Examined, 2002.

[7] Faja S., "Pair Programming as a Team Based Learning Activity: A Review of Research," *Issues in Information Systems*, vol. 12, no. 2, pp. 207-216, 2011.

[8] Hannay J., Arisholm E., Engvik H., and Sjoberg I., "Effects of Personality on Pair Programming," *IEEE Transactions on Software Engineering*, vol. 36, no. 1, pp. 61-80, 2010.

[9] Kavitha R. and Irfan A., "Knowledge Sharing Through Pair Programming in Learning Environments: An Empirical Study," *Education and Information Technologies*, vol. 20, no. 2, pp. 319-333, 2015.

[10] Kuppuswami S. and Vivekanandan K., "The Effects of Pair Programming on Learning Efficiency in Short Programming Assignments," *Informatics in Education-An International Journal*, vol. 3, no. 2, pp. 251-266, 2004.

[11] Lui K., Keith C., and Chan C., "Pair Programming Productivity: Novice-Novice vs. Expert-Expert," *International Journal of Human-Computer Studies*, vol. 64, no. 9, pp. 915-925, 2006.

[12] Muller M., "Two Controlled Experiments Concerning the Comparison of Pair Programming to Peer Review," *The Journal of Systems and Software*, vol. 78, no. 2, pp. 166-179, 2005.

[13] Naufal I. and Hui T., "Learning Style, Metaphor and Pair Programming: Do they Influence Performance?," *Procedia -Social and Behavioral Sciences*, vol. 46, pp. 5603-5609, 2012.

[14] Pikkarainen M., Haikara J., Salo O., Abrahamsson P., and Still J., "The Impact of Agile Practices on Communication in Software Development," *Empirical Software Engineering*, vol. 13, no. 3, pp. 303-337, 2008.

[15] Pressman S., *Software Engineering Practitioner's Approach*, McGraw Hill, 2010.

[16] Radermacher A. and Walia G., "Gaps Between Industry Expectations and the Abilities of Graduates," *in Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, Colorado, pp. 525-530, 2013.

[17] Salleh N., "A Systematic Review of Pair Programming Research-Initial Results," *in Proceedings of New Zealand Computer Science Research Student Conference*, Christchurch, pp. 151-158, 2008.

[18] Salleh N., Mendes E., and Grundy J., "Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review," *IEEE Transactions on Software Engineering*, vol. 37, no. 4, pp. 509-525, 2011.

[19] Scott G. and Wilson D., "Tracking and Profiling Successful IT Graduates: An Exploratory Study," *in Proceedings of the 13th Australasian Conference on Information Systems*, Australasian, pp. 1185-1195, 2002.

[20] Vafadar S. and Barfourosh A., "Towards Intelligence Engineering in Agent Based Systems," *The International Arab Journal of Information Technology*, vol. 12, no. 1, pp. 94-102, 2015.

[21] Vanhanen J. and Korpi H, "Experiences of Using Pair Programming in an Agile Project," *in Proceedings of the 40th Hawaii International Conference on System Sciences IEEE Computer Society*, Waikoloa, pp. 1530-1605, 2007.

[22] Venkatesan V. and Sankar A., "Adoption of Pair Programming in the Academic Environment with different Degree of Complexity in Students Perspective-An Empirical Study," *International Journal of Engineering Science and Technology*, vol. 2, no. 9, pp. 4791-4800, 2010.

[23] Williams L. and Kessler R., *All I Ever Needed to Know about Pair Programming I Learned in Kindergarten*, Communications of the ACM, 2000.

[24] Williams L. and Kessler R., *Pair Programming Illuminated*, Addison Wesley, 2003.

Kavitha Karthiekheyan received her Master of Computer Applications Degree and M.Phil Degree from Bharathiar University, Coimbatore, India. Currently she is working as an Assistant Professor in the Department of MCA in Kumaraguru College of Technology, Coimbatore. She has 15 years of teaching experience and 4 years in research. She has published research papers in various international conferences and journals.

Irfan Ahmed holds his PhD degree in computer science from the Alagappa University, Karaikudi, India. He has more 2 years of industry experience, 18 years of teaching experience and 10 years of research experience. Currently he is working as Director in the Department of MCA in Sri Krishna College of Engineering and Technology, Coimbatore, India. He received the best faculty award in the year 2012. He has presented and published more than 20 research papers in international journals and conferences.

Jalaja Jayalakshmi received her Master of Computer Applications Degree from Bharathiar University, Coimbatore, India. Currently she is working as an Assistant Professor in the Department of MCA in Kumaraguru College of Technology, Coimbatore, India. She has 10 years of experience in teaching and 1 year in industry.