VOL. 10, NO 22, DECEMBER, 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

KNOWLEDGE SHARING THROUGH INTERNALIZATION PROCESS OF PAIR PROGRAMMING IN STUDENT'S ACADEMIC PROJECTS

V. Venkatesan and A. Sankar
Department of Computer Applications, PSG College of Technology, India
E-Mail: v_venkkatesan@yahoo.co.in

ABSTRACT

Students continue to struggle with learning to program, for reasons that the hypothesis is not solely cognitive. This study contributed for better understanding of important knowledge sharing activities to construct student's learning skills during internalization process through pair programming. The pair programming is one of the very important pedagogical approaches that can enhance students' abilities in the area of computer programming. Knowledge sharing in pair programming can be improved with the guidance of lecturers or teaching assistants and also increasing the frequency of programming activities between the student pair members. In the academic setup, based on few set of experiments, we found that students who used pair programming produced better programs, confident in their solutions, and enjoyed completing the assignments more than students who programmed alone. Pair programming improves the design ability, reduces time taken to do their academic mini-projects and it increases the knowledge and programming skill of the pair members. This experimental study represents the results of the knowledge sharing during pair programming exercise carried out with sixty Master of Computer Applications students who are engaged in developing small applications as a part of their Mini-project during their II year at PSG College of Technology, during the odd semester of 2014.

Keywords: Pair programming, extreme programming, knowledge sharing.

1. INTRODUCTION

In recent past years the concept of pair programming has evolved as one of the important technique of coding. Extreme Programming (XP) is a computer software development approach that credits much of its success to the pair programming by all programmers, regardless of experience (L. Williams, 2000; Cockburn.A. *et al*, 2000). Extreme Programming focuses on disseminating knowledge through collaborative practices such as Pair Programming where teams of two people work together, Test Driven Development which is to writing lots of tests, and writing them early, Continuous Integration represents putting code together as soon as code written, not at the last minute, Coding Standards which helps to learn and follow well established conventions, Collective Code Ownership where members are responsible for the partner's code and do Simple Design.

The idea behind these practices is to sharing of knowledge, work and to expertise with all teams (Beck. K, 2000). XP promotes testing code literally every few minutes, after every minor change in code. It works best for relatively small projects size with a small number of efficient programmers. It is assumed that while programming in pairs, people try harder to code well and thus gains confidence in programming and knowledge.

2. PAIR PROGRAMMING

Pair programming is a software development technique where two programmers work together side-by-side on the same machine to achieve their goals. In this programming methodology two programmers collaborate on the same programming task at the same time, with each programmer having their distinct role and responsibilities. The primary role is called the driver; this is the programmer with control of the input devices, usually a single keyboard and mouse combination, and there are actively coding the solution to the task faced by the pair. The other programmer has the role of navigator, and this programmer has the responsibility to review the driver's work on as much level as they can. "Tactical defects" is a term for low level mistakes, such as syntax errors, types and basic logic errors (Williams L. *et al*., 2002). "Strategic defects" is the term for high level errors, when the solution being implemented heads in the wrong direction or has non-local effects on the system.

It is responsibility of the navigator to consider both tactical and strategic defects and attempt to mitigate them as the driver produces them while coding (Saurabh Ratti, 2008).

Most of the previous researches have evaluated the effectiveness of pair programming by measuring the characteristics of the product (computer programs) that were developed by the students. Each of the products was delivered by a pair. However, to prove that pair programming is an effective learning methodology, it should also improve the individual students' subject knowledge and programming skills. Only, very few attempts have been made to measure the individual performance.

Software design plays an important role in software development in industry. The same is also true

for developing programs for student projects since, it is also used for evaluation in educational institutions.

When pair programming is used in long duration projects, the students have to work together both at educational institutions and at home. In this kind of environment, the students had reported that it was difficult to find time to meet and work together.

Various studies have been done on determining the usefulness and effectiveness of Pair Programming as pedagogical tool and indicated the following positive results: i) Pair programming can improve students' performance by gaining higher scores on programming assignment (Werner *et al*., 2004; McDowell *et al*., 2003; Cliburn, 2003; Slaten *et al*., 2005). ii) Pair programming can increase student's confidence and satisfaction (Werner *et al*., 2004; McDowell *et al*., 2003; Cliburn, 2003; Slaten *et al*., 2005). ii) Pair programming can encourage students to complete the programming course (Werner *et al.*, 2004; McDowell *et al*., 2003). The studies suggested that pairing might be impractical when deadlines are tight. (Paranjape, 2001; Becker-Pechau, Breitling *et al*. 2003; Gittins, Hope *et al*. 2001) all suggest that pairing could more useful when it is introduced in a non-mandatory fashion, perchance focusing its use on a most critical or complex tasks.

Some Studies have shown that pair programming creates an environment conducive to more advanced, active learning and social interaction, leading to students being less frustrated, more confident, and more interested in Information Technology. Pair programming encourages students to interact with peers in their classes and laboratories, thereby creating a more communal and supportive environments.

2.1 Advantage of pair programming

Earlier task completion: Pair programming can potentially increase the development speed and thus shorten time-to-market. On the economics' perspective, two developers working in pairs is only superior when the resulting time is less than halved compared to individual all programming.

Better code readability: Readability promotes exchangeability of code and potentially lowers program maintenance cost. Related evidence such as improved conformance of coding standard.

Early identification of faults: Due to continuous code review, potential software faults can be identified earlier, where as in solo programming, a fault typically exists until an appropriate code review or test is undertaken.

2.2 Integrating pair programming into a software development process

Programming pairs routinely "refactor" the code base by continuous change and enhancement. They view the code as the self-evolving design- they do not spend time on a design document.

One of the things that we firmly believe in is the value of collaboration and learning from the each other, especially when programming.

In pair programming at its simplest, a pair of students will work together on a programming assignment.

The two programmers periodically switch their roles; they work together as equal to develop software. This practice of pair programming can be integrating into any software development process.

2.3 Problem solving

Groups of students work together during the problem solving process and regularly called on to present their solutions on the entire class. Students who get the most out of these workshops [Donna Teague *et al*. 2007] tend to be those that are actively involved in discussions during the problem solving process.

Novice programmers require hands-on experience, and lots of it, because their knowledge of programming is not passively absorbed through texts and lectures, but rather actively constructed via their own practical experiences. Students should be given a supportive environment in which to experiment, and get the practical experience they need. Providing a totally collaborative learning (Donna Teague *et al*. 2007) environment may provide the support that students need to develop sound problem solving and programming skills.

2.4 Pair learning

Knowledge is constantly being passed between partners, from tip on tool usage, to programming language rules, design and programming idioms, and overall design skill. Pair programming may contextualize the learning activity in a manner that allows the students to focus on the different knowledge types, and provide the feedback necessary to increase their ability to develop monitoring mechanisms for their own learning activities (Furberg *et al.*, 2013). If the pair can work together, then they learn ways to communicate more easily and they communicate more often with each other.

Existing literature on the pedagogical adoption of pair programming indicate several benefits like enhanced learning, knowledge transfer and improvements in student performance.

One significant benefit to pair programming is that it is a common professional practice situated in the workplace. Therefore, exposure to pair programming in the classroom could familiarize students with a method they will be required to employ on the job (Madhumita Singha Neogi *et al*, 2011). Similarly, pair programming could supplement existing curricula with additional learning opportunities that could support students in future careers. For example, students might learn method for collaboratively solving problems, communicating their

understanding verbally, and resolving disagreements with their peers.

This study focuses on the quality of the program developed by students using pair programming method, their performance and confidence towards developing a program using coding standard, code efficiency, testing and documentations (Subbaraya Kuppuswami and Kalimuthu Vivekanandan, 2004). Also this research focuses on the knowledge transfer of the students during programming.

3. KNOWLEDGE SHARING IN ACADEMIC PROJECTS

Pair programming raises issues for project management. On the positive side, it may act as a backup for absent or departing developers [Williams, L., *et al*, 2000]. This was also reflected in the some literature with 'no one person has a monopoly on any one section of the code, which should remove organizational dependencies on particular resources and mitigate risk to the businesses. On a less positive note, there are challenges for project management in terms of planning and estimation. This was highlighted in the study by 'Methods of planning/estimating need to change when team is pair-programming rather than tackling tasks as individuals' Knowledge sharing and transfer is one of the most widely-claimed benefits of pair programming [Mawarny *et al*, 2011]. An experiment was conducted in PSG College of Technology, with 60 students of II year Master of Computer Application (MCA) for their Mini Project-I course during the odd sem 2014.

The mini project was evaluated in three stages, say project review1, project review2 and final project viva with demo. For this, 25% of marks will be allotted for review one and for second review another 25% of marks and for final project viva and demo 50% marks will be evaluated. The entire project review will be done under various faculty review panels and concerned faculty guide.

Students are advised to choose a domain for doing their project. The class tutor will form pairs based on the domain of the students and then faculty guide be assigned. During First two review students are allowed to present project presentations with their partners. But for the final project demo and viva they must face the review panel individually. In review1 each student explained about the project objective, system analysis and design, their roles, tools and technologies to be used, implementations and deadlines of the project. From day one of the project work each students help each other and to share their knowledge to complete project in time. During second review, students were required to partially complete their project utilizing the pair programming practice.

The students frequently updated their project status to their faculty guide/ project guide. They worked with the same partner for entire project until completed.

In this experiment, out of 60 students 40 were programming their project in paired group and 20 students did their projects in solo programming. Our study was specifically aimed at the analysis of pair programming on both students groups toward knowledge sharing and quality of work. The Figure-1 shows the flow diagram of the evaluation process of academic mini-project work.

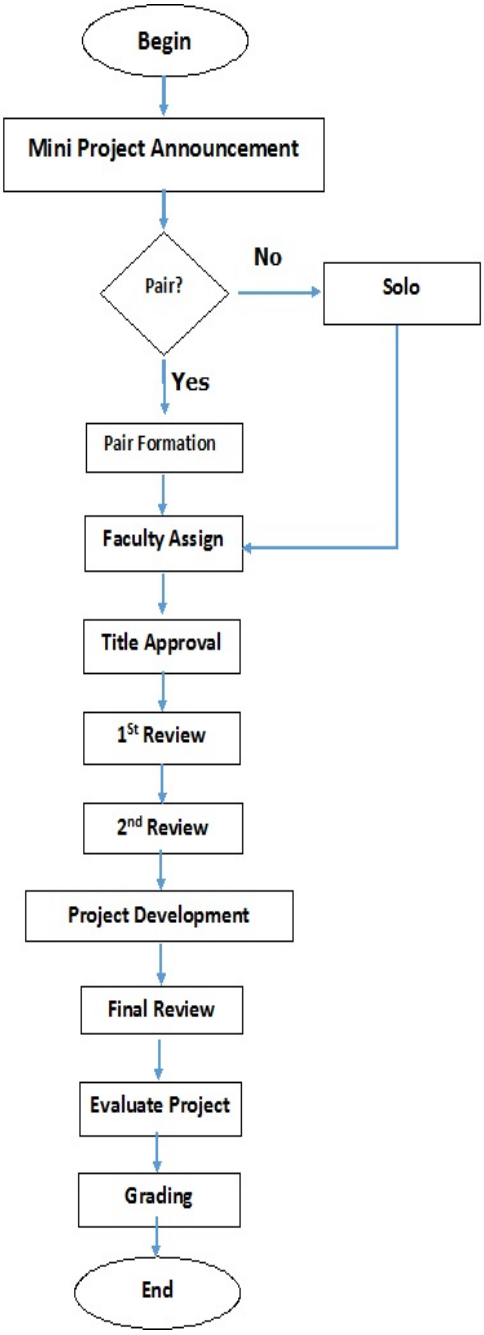

Figure-1. Pair programming process in mini project.

4. PAIR PROGRAMMING ANALYSIS

As discussed earlier, the data sources, collection procedures and tools evolved during the mini project. This is why all metrics have not been calculated for each project. Each metric is defined in the subsequent section together with the empirical results. Table-1 shows the data table of coding standard where the difference in the coding standard and the corresponding chart shown in Figure-2.

Hypothesis is there was significance difference on improvement of Coding Standard in Pair programming with the following value Pair Mean: 75, SD: 35 and Solo Mean: 54, SD: 11 at Level of significance α=5% using t-distribution.

Table-1. The Comparisions of solo and pair programming in terms of coding standard.

Percentage	0%	25%	50%	75%	100%
Solo	1	6	8	8	2
Pair	0	3	7	13	13

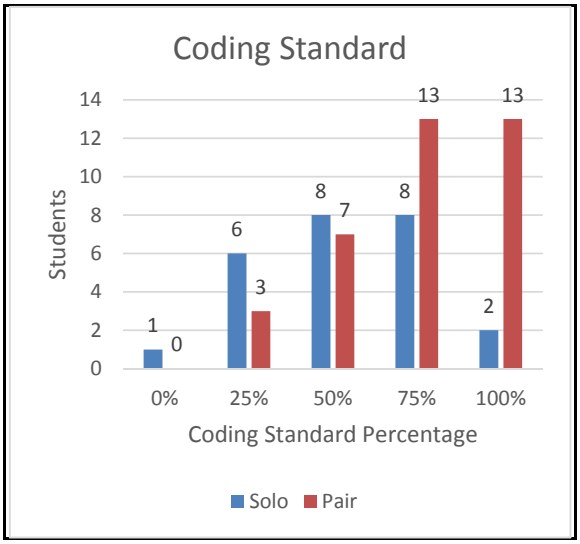

Figure-2. Coding standard.

Table-2 shows the efficiency data table where the difference in the Efficiency is shown in Figure-3.

Hypothesis is there was significance difference on improvement of efficiency in Pair programming with the following value Pair Mean: 75, SD: 36 and Solo Mean: 64, SD: 19 at Level of significance α=5% using t-distribution.

Table-2. The comparisions of solo and pair programming in terms of efficiency.

Percentage	0%	25%	50%	75%	100%
Solo	0	6	5	8	6
Pair	0	2	7	15	12

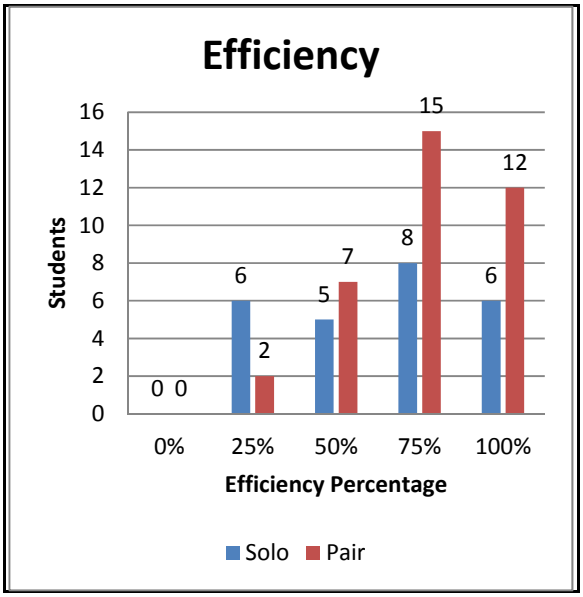

Figure-3. Efficiency.

Table-3 shows the testing data table where the difference in the testing is shown in Figure-4.

Hypothesis is there was significance difference on improvement of testing in Pair programming with the following value Pair Mean: 82, SD: 45 and Solo Mean: 58, SD: 16 at Level of significance α=5% using t-distribution.

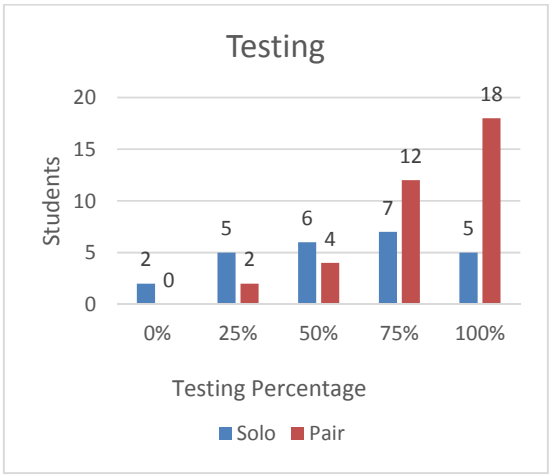

Figure-4. Testing.

Table-3. The comparisions of solo and pair programming in terms of testing.

Percentage	0%	25%	50%	75%	100%
Solo	0	5	6	7	5
Pair	0	2	4	12	18

Table-4 shows the data table of Documentation where the difference is shown in Figure-4.

Table-4. The comparisions of solo and pair programming in terms of documentation.

Percentage	0%	25%	50%	75%	100%
Solo	0	4	8	8	5
Pair	0	2	4	14	16

Hypothesis is there was significance difference on improvement of documentation in Pair programming with the following value Pair Mean: 80.55, SD: 42 and Solo Mean: 64, SD: 18 at Level of significance α=5% using t-distribution.

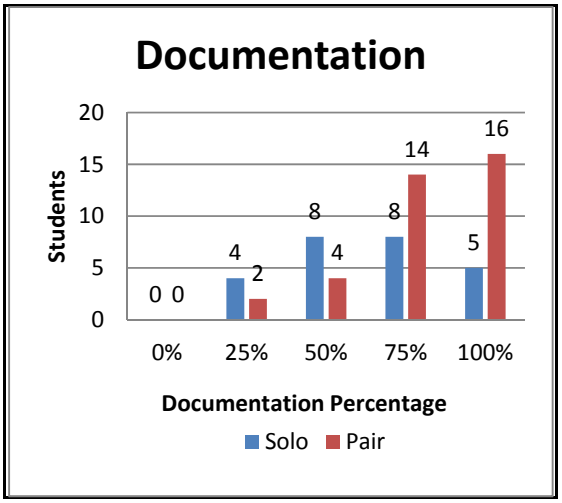

Figure-5. Documentation.

These metric describing the actual use of pair programming in the students projects is productivity, which provides information on how pair programmer's productivity evolves as the project progresses and also allows comparing the productivity of the two different programming styles. In this study, productivity is calculated for both pair and solo programming styles for each iteration as a ratio of produced logical code lines and spent effort. It is acknowledged that measuring productivity is not a straight forward task and using lines of code counts has its challenges. However, it is the most commonly used means for describing productivity and thus used also here. The numbers of code lines for each programming style were obtained by calculating the amount of code lines in the iteration's end baseline made with each programming style. The total productivity of pair and solo programming in the three sections on mini projects. There seems to be no regularity between the productivity of different programming styles: in case two solo programming has a bit higher productivity than pair programming, in case three the situation is reversed, and in case four, pair programming has substantially higher productivity than solo programming.

4.1 Rationale for pair programming

The results concerning with the rationale for pair programming obtained through team surveys are presented in the following. The focus of this qualitative data (i.e., recorded, described) is on determining the types of tasks and situations, which developers find especially suitable or unsuitable for pair programming [H. Hulkko. *et al*., 2005]. The study aimed at collecting team members' views about the usefulness of pair programming in different application situations and development phases. One developer found pair programming to be suitable for many coding tasks, but not necessarily like installation tasks. The team members of cases one, three and four pair programming to be especially useful for novice team members and in the beginning of a project.

The effect of the complexity of the task on the usefulness of pair programming was also brought up by the developers in the final interviews. The developers felt that pair programming was more useful for demanding and complex tasks than for role tasks.

The first metric used to describe the quality effects of pair programming is related to adherence to coding standards. In accordance to agile philosophy, the project team was responsible for defining the coding standards in the beginning of each project, and the code has been compared against these same standards when deriving this metric.

5. COMMUNICATION

Communication plays an important role during the planning game and when pair programming. In the planning game story cards help to structure the communication. Pair programming and testing rely on a flexible, unstructured and creative communication between pair members where communication is guided by the source code the pair is working on. For an example, the possible channel of communication is shown in Figure-6.

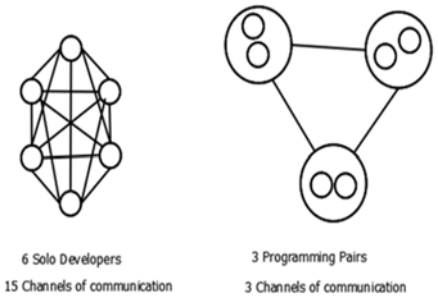

Figure-6. Communication channel.

These meetings acted as a communication channel between the team and the organization. The practice communicate pair had a positive effect on the collaboration between the team and its surrounding organization since it provided the organization and the team with means for discussing software development issues on a more detailed level. Everyone needs to be in touch with each other throughout the day.

Software projects require a great deal of communication. If you're writing an application just for your own use, then the communication channels are all extremely fast, making for very tight feedback loops.

Solo developers can inter communicate n members they need = n*(n-1)/2 way. So they need more time to finish their work. But pair programmer need only minimum communication n pair = n communication way they used.

6. PRODUCT QUALITY

Pair Programming is not just about producing the actual code of programming. It also collaborating as pairs with tasks as: analysis, architecture and low-level design, test or other software related problems affecting the software and its quality. Selected metrics were used as key performance indicators to assess the software quality on regular basis. This information was used for making the internal software quality transparent for management. Additionally, explicit tasks for quality improvement have been filed to address and schedule software quality issues within pair development process. The students performed much more consistently and with higher quality in pairs than they did individually - even the less motivated students performed well on the programming projects.

The students felt they were more productive when working collaboratively. They were several reasons observed. First, when they met with their partner they both worked very intently each kept the other on task and were highly motivated to complete the task at hand during the session. The Table-5 shows positive aspects of pair programming based on the data collected from students.

Table-5. Positive aspects of pair programming.

Variables	Mean
Learning	3.92
Quality of work	4.59
Knowledge sharing	4.51
Responsibility	4.11
Pair support	3.87
Roles Switching	3.09
Usefulness	4.57
Satisfaction	4.42
Overall Productivity	4.15

The pair students were extremely positive about their collaborative experience. The Figure-7 shows the positive aspects of pair programming based on the data given be pair members.

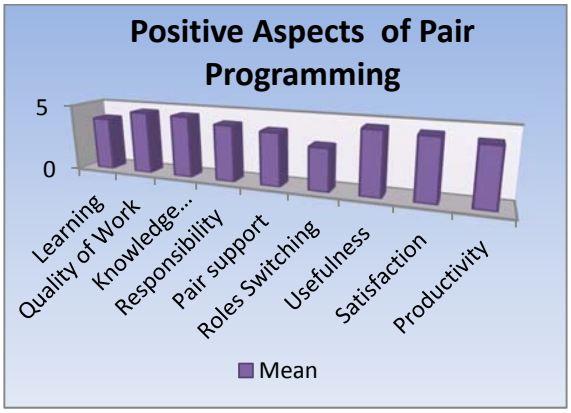

Figure-7. Positive aspects of pair programming.

All pair Members was happier and less frustrated with the class. They also felt good and able to come up with more creative, efficient solutions when working with a partner.

7. CONCLUSIONS

Pair programming would be the key benefit as this would enable students to share mentoring roles, reflect on their own and each other's work and share the learning experience together during problem solving and program development. Programming really is complex and difficult to learn, there also cultural and social influences on students presenting to introductory computer science courses. Students' knowledge building, reflected in scores, contributed to their domain understanding. This paper highlights the advantages of intensive collaboration between students by exploiting the students' own ability and desire to interact with their peers. Peer interaction can

lead to very strong learning experiences and overall there is a good knowledge sharing among students. Our findings suggest that not only does pairing not compromise students' learning, but that it may enhance the quality of their programs.

REFERENCES

[1] Beck. K. 2000. Extreme Programming Explained (Addison-Wesley.

[2] Cliburn D.C. 2003. Experiences with pair programming at a small college. Consortium for Computing Science in College. 19(1): 20-29.

[3] Cockburn. A and L. Williams. 2000. The Cost and Benefits of Pair Programming. 1st International Conference of Extreme Programming and Flexible Processes in Software Engineering, Italy.

[4] 2007. Donna Teague and Paul Roe: Learning to Program: Going Pair-Shaped. Innovation in Teaching and Learning in Information and Computer Sciences. 6(4).

[5] Furberg A., Kluge A. and Ludvigsen S. 2013. Student sense making with science diagrams in a computer-based setting. International Journal of Computer-Supported Collaborative Learning. 8(1): 41-64.

[6] Hulkko. H and Abrahamsson. P. 2005. A Multiple Case Study on the Impact of Pair Programming on Product Quality. In International conference on Software Engineering (ICSE).

[7] Madhumita Singha Neogi and Vandana Bhattacherjee. 2011. Pair vs Solo Programming: Students' Perceptions. International Journal of Computer Science and Information Technologies. 2(3).

[8] Mawarny Md. Rejab, Mazni Omar, Mazida Ahmad, Khairul Baraih Ahamad. 2011. Pair Programing in inducing knowledge sharing. ICOCI, pp. 11-20.

[9] McDowell C., Hanks B. and Werner L. 2003. Experimenting with pair programming in the classroom. Proceedings of the 8th annual conference on innovation and technology in computer science education. 35(3): 60-64.

[10] McDowell C., Werner L., Bullock, H.E. and Fernald J. 2003. The impact of pair programming on student performance, perception and persistence. Proceedings of the 25th International Conference on Software Engineering. pp. 602-607.

[11] Saurabh Ratti. 2008. Pair Programming - Software Engineering Project Management Project Report.

[12] Slaten K.M., Droujkova M., Beenson S.B., Williams L. and Layman L. 2005. Undergraduate student perceptions of pair programming and agile software methodologies: verifying a model of social interaction. Proceedings of the Agile Development Conference. Software Engineering. 36: 61-80.

[13] Subbaraya Kuppuswami and Kalimuthu Vivekanandan. 2004. The Effects of Pair Programming on Learning Efficiency in Short Programming Assignments. Informatics in Education. Vol. 3.

[14] Werner L.L., Hanks B. and McDowell C. 2004. Pair-programming helps female Computer Science Students. ACM Journal of Educational Resources in Computing. 41(3).

[15] Willams. L and R. Kessler. 2003. Pair Programming Illuminated. Addison- Wesley.

[16] Williams L. *et al*. 2006. Examining the Compatibility of Student Pair Programmers. Agile Conference 2006, Minneapolis, MN. pp. 411-420.

[17] Williams, L. *et al.* In Support of Pair Programming in the Introductory Computer Science Course. Computer Science Education. 12: 197-212.

[18] Williams L.A. and Kessler R.R. 2000. The effects of "pair-pressure" and "pair-learning" on software engineering education. Thirteenth Conference on Software Engineering Education and Training. pp. 59-65.

[19] Williams. L and. Kessler R. R. 2002. Pair Programming Illuminated, Boston, MA: Addison-Wesley.

[20] Williams. L, *et al*. 2000. Strengthening the Case for Pair Programming. In IEEE Software.