ISSN:0975-9646

Pair vs Solo Programming: Students' Perceptions

Madhumita Singha Neogi[1] , Vandana Bhattacherjee[2]
[1]*Deptt. of IM, Xavier Institute of Social Service, Ranchi*
[2]*Deptt. of CS & E, Birla Institute of Technology, Ranchi*

Abstract **-In recent past the concept of pair programming has evolved as one of the important technique of coding. It is one of the most talked about aspects of XP. Pair programming is a methodology in which two people work together and periodically switch between the roles of driver and navigator. Instead of partitioning a task into a number of activities, where each member performs a different activity alone, in pair both partners perform each activity together. Pair programming concepts have been introduced in the classroom and students' reaction to the same has been presented in this paper. Further, this paper presents the results of an experiment conducted to assess the pair programmers' as well as individual programmers' ability in terms of effort and efficiency. The study was conducted with two groups of students. Both the groups solved the lab assignments in pair as well as in solo programming technique. The two parameters were evaluated in this study one is the effort and the other is the efficiency. Effort was measured as number of hours spent per person and efficiency we have measured as number of test cases passed after completion of the assignments as well as number of failures per ten runs. The students were given a questionnaire after completion of the assignments and evaluation. Their responses were evaluated using factor analysis and performance of pair and solo programming technique were compared using paired sampled t-test. This paper presents the results of the factor analysis and t-test. The results show that pair programming technique has greater impact than solo programming in terms of effort and efficiency.**

Keyword **-Pair programming, solo programming, effort, efficiency.**

1. INTRODUCTION

A key concept during the coding activity is pair programming. It is one of the most talked about aspects of XP. Pair programming is a methodology in which two people work together and periodically switch between the roles of driver and navigator. Instead of partitioning a task into a number of activities, where each member performs a different activity alone, in pair work, both partners perform each activity together. In this research work, pair programming concepts were introduced in the classroom and students' reaction to the same has been presented. This paper presents the results of a study conducted to assess the pair programmers' as well as individual programmers' ability in terms of effort and efficiency. Effort is measured in terms of time spent to complete the job and efficiency in terms of number of acceptance test cases passed. These two key issues are studied through the controlled experiment by pair programmers and individuals. The data was collected from the students through a questionnaire based on five scales. The questions were related to solo and pair programming as well as effort and efficiency. The data was then factor analyzed through a statistical package.

2. RELATED Work

Carver et al. have conducted a study on increased retention of early computer science and software engineering students using pair programming. The results of the study showed that retention significantly increased for those students already majoring in Computer Science, Software Engineering, or Computer Engineering. In addition, survey result indicated that the students viewed many aspects of pair programming to be very beneficial to their learning experience (Carver J.C., 2007). In another study Slaten et al. studied undergraduate student perception of pair programming and agile software methodologies. According to them one of the reasons that undergraduate students, particularly women and minorities, can become disenchanted with computer science education is because software development is wrongly characterized as a solitary activity. The finding suggest that pair programming and agile software methodologies contribute to more effective learning opportunities for computer science students and that students understand and appreciate these benefits (Slaten et al., 2005). Muller et al. have compared the program defects caused by pair programmer and solo programmer. Their assumption was that pair programmer makes few mistakes than solo programmers regardless of the programmer task and defect type (Muller, M.M., 2007).

Erik Arisholm et al studied the effects (duration, effort, and correctness of the maintained program) of pair programming versus individual programming, and concluded that it will depend on the moderating variables system complexity and programmer expertise, both of which will have an impact on the perceived complexity of programming tasks (Arisholm, E., 2007). A study has been conducted by Vivekanandan et al to investigate the students' attitude on important issues of pair programming. The opinion of students were obtained on pair programming after asking them to take up pair programming to do laboratory exercises. The data were analyzed, results indicate that the students like to adopt pair programming as a learning methodology. They also like to have partners whose academic achievement is same or higher (Vivekanandan, K., 2007). Kyungsub S. et al have studied the concept of pair programming in another way. Their paper focuses on the effects of some psychosocial factors, a programmer's personality type, may have on the pair programming environment (Kyungsub, S., 2007).

Dyba et al have presented a study on effectiveness of pair programming. They show the Meta-analyses of pair programming effects on (a) quality, (b) duration, and (c) effort (Dyba, T., 2007). Succi et al. propose an experimental framework to quantify benefits and costs of the pair programming practice and compare design aspects of the resulting software products and their defect behavior. For this purpose, they use a set of object-oriented metrics and software reliability growth models based on occurrence of service requests (Succi, G., 2001). Canfora et al. report the findings of a controlled experiment on pair programming, applied to the design phase and performed in a software company. The results of the experiment suggest that pair programming slows down the task, yet improves quality (Canfora, G., 2006).Theodore et al. evaluated the usefulness of pair programming in a classroom setting (Toll III, T.V., 2007). A two-phased study of 1350 students was conducted by Williams et al. at North Carolina State University from 2002-2005 to determine if teaching staff can proactively form compatible pairs based upon any of the following factors: personality type, learning style, skill level, programming self esteem, work ethic, or time management preference. They examined compatibility among freshmen, advanced undergraduate and graduate student pair programmers. They have found that overall 93% of students are compatible with their partners (William, L., 2006).

Vanhanen et al. present experiences of using PP extensively in an industrial project (Vanhanen, J., 2007). Succi et al. in their paper reports the preliminary results of an analysis of the effects of pair programming on job satisfaction. A questionnaire on pair programming techniques has been compiled and posted on the web. 108 responses have been collected from around the world. The preliminary results evidence a very positive effect of pair programming on job satisfaction (Succi, G., 2005).

Kwak, Y. et al. in their paper addresses lessons learned from implementing project risk management practices in software development environment (Kwak, Y., 2005). Lui et al. explore the efficacy of pairs versus individuals in program design-related tasks separately from coding (Lui, K.M., 2008).

Authors have also conducted studies in several University settings regarding student software development patterns (Bhattacherjee, 2007, 2008, 2009a, 2009b, 2010) (Neogi, 2007, 2009a, 2009b, 2010).

3. OBJECTIVE OF THE STUDY

The goal of the study is to investigate and compare pair and solo programming technique in lab environments and their effects on few parameters of software. In this paper we have given main emphasis on the comparison of effort and efficiency on pair and solo programming. Our previous studies have shown that students prefer group work in solving assignments.

The objective of this study is to address the following questions:

RQ1 *Is pair-programming technique better than solo-programming?*

Null hypothesis in our study was that there is no significant difference in pair-programming technique and solo-programming. It can be stated as

- ***H_{01}:μ_d=0*** where μ_d is the difference of mean (difference between preferring pair programming technique and solo programming technique)

Alternate hypothesis was that difference between pair-programming and solo programming is not equal to zero. It can be stated as

- ***H_{11}: $\mu_d \neq 0$*** i.e. $\mu_{pair(Programing-technique)} \neq \mu_{solo(Programing-technique)}$

RQ2 *Is pair-programming effective in reducing the software development effort compared to solo-programming?*

Null hypothesis in our study was that there is no significant difference in development effort between pair-programming and solo-programming. It can be stated as

- ***H_{02}:μ_d=0*** where μ_d is the difference of mean

Alternate hypothesis was that mean effort for development using pair-programming is not equal to solo-programming. It can be stated as

- ***H_{12}: $\mu_d \neq 0$*** i.e. $\mu_{pair(effort)} \neq \mu_{solo(effort)}$

RQ3 *Is pair-programming effective in improving the efficiency of programs as compared to solo-programming?*

Null hypothesis in our study was that there is no significant difference in efficiency between the pair-programming and solo-programming. It can be stated as

- ***H_{03}:μ_d=0*** where μ_d is the difference of mean

Alternate hypothesis was that mean efficiency of programs using pair-programming is not equal to solo-programming. It can be stated as

- ***H_{03}: $\mu_d \neq 0$*** i.e. $\mu_{pair(efficiency)} \neq \mu_{solo(efficiency)}$

4. EXPERIMENT DESCRIPTION

4.1 Metrics Used

The experiment was conducted on the computer programming lab. While given assignments the researcher observation was on three key parameters. These are programming technique, effort and effectiveness on pair and solo programming.

Programming Technique: Here we have given emphasis on implementing pair programming technique among students while solving lab assignments and compared their opinion with solo programming technique which they normally follow while solving lab assignments. Pair programming technique where two persons sitting together share a common machine; while one person writing code the other guides the person writing code. The first person called the driver and the second person is called the navigator in pair programming technique. Solo programming is the technique where single person using a machine writing the code nobody involve in guiding him while writing code.

Effort: The total effort in person-hour to complete the project. The data collected was number of hours counted after the end of twelve consecutive weeks. The total numbers of hours spent for each program was noted by each group of students. We have measured effort as total time spent to develop the projects including all the phases. Nonproductive time between the tasks was not included. The total effort for the pairs was thus the duration for the pair multiplied by two.

Effectiveness: Percentage of features implemented measured as the percentage of acceptance test cases passed.

A. Test Exercise

A structured interview schedule was prepared in English. The questionnaire initially contained 30 items. The questionnaire was scrutinized by 3 experts and item agreed by all were retained in the final questionnaire. The final questionnaire consisted of a scale of 19 items. Each item was measured in a Likert type 5 time scale – (i) Not at all, (ii) slightly, (iii) moderately, (iv) highly, (v) very highly.

When scores of all items on total sample were factor analyzed using principal component analysis and the factors were rotated through varimax procedure, two factors were extracted which explained 41.7 % of total variance (See appendix A). The first factor had significant loadings on 10 items related to 'Pair Programming'. The sample items include "Do you think pair programming improves effective learning?" and "Do you think using pair programming technique, efficiency of coding has improved?" etc. The second factor had loadings on 9 items which assessed 'Solo Programming". The sample items include "Do you think you would have paid more attention on programming technique using solo programming/" and "Do you think you would not try to avoid responsibility if you did not have a partner?" etc.

Alpha reliability on the current sample was 0.86 for pair programming and 0.75 for solo programming. High score indicated (pair/ solo) more efficiency or more preference for the particular programming.

B. Pair programming

Pair programming was measured with a set of 10 questions. As mentioned earlier that each item was measured in a Likert type 5 time scale – (i) Not at all, (ii) slightly, (iii) moderately, (iv) highly, (v) very highly.

When scores of all items on total sample were factor analyzed using principal component analysis and the factors were rotated through varimax procedure, two factors were extracted which explained 53.56 % of total variance (See appendix B). The first factor had significant loadings on 5 items related to 'Pair Effort'. The sample items include "Do you think pair programming improves effective learning?" and "Do you think pair programming has improved your ability to critically analyze?" etc. The second factor had loadings on remaining 5 items which assessed 'Pair Efficiency". The sample items include "Do you think using pair programming technique, efficiency of coding has improved?" and "Do you think working with your partner in the same machine improves quality of code" etc.

Alpha reliability on the current sample was 0.77 for pair effort, and 0.75 for pair efficiency. High score indicated (pair time/ pair effort) more efficiency or more preference for the particular programming.

C. Solo programming

Solo programming was measured with a set of 9 questions. As mentioned earlier that each item was measured in a Likert type 5 time scale – (i) Not at all, (ii) slightly, (iii) moderately, (iv) highly, (v) very highly.

When scores of all items on total sample were factor analyzed using principal component analysis and the factors were rotated through varimax procedure, two factors were extracted which explained 50.90 % of total variance (See appendix C). The first factor had significant loadings on 7 items related to 'Solo Effort'. The sample items include "Do you think you would have learned more if you had developed the assignments alone?" and "Do you think you would have solved the assignments alone in less time than using pair programming?" etc. The second factor had loadings on remaining 2 items which assessed 'Solo Efficiency'. The sample items include "Do you think you would have learn more through pair programming if your partner was of less academic level than yours?" etc.

Alpha reliability on the current sample was 0.76 for solo effort, and 0.50 for solo efficiency. High score indicated (pair time/ pair effort) more efficiency or more preference for the particular programming.

The variables calculated for this study

Pair=Total number of weighted questions related to pair /10
Solo= Total number of weighted questions related to solo/9
PEffort= Total number of weighted questions related to pair effort/5
SEffort= Total number of weighted questions related to solo effort /7
PEfficiency= Total number of weighted questions related to pair efficiency/5
SEfficiency= Total number of weighted questions related to solo efficiency/2

4.2 Procedures

The experiment was conducted during the lab classes. First, the subjects were given an introduction to the experiment by the researcher in the classroom sessions. The subjects were also introduced to the concept of Pair Programming (PP) during the software engineering theory classes. There it had been discussed at length about active collaboration in PP and about two roles (driver and navigator). The subjects were allotted one machine per pair and were told to switch roles during lab classes. The lab sessions were conducted under supervision of the researcher.

4.3 Subjects

The subjects were chosen from two different colleges. A group of computer science students was considered for the experiment from an Engineering college of West Bengal and the other group of post graduate students undergoing a course of MBA systems specialization at Ranchi. Class strength is of 60 each. Few students were absent on the day the questionnaire was collected. By the time of the experiment all subjects had already gone through two programming languages. The subjects were given assignments of Visual basic and Java programming language. All the subjects were supposed to do the assignments in alone and in pair as per the mentioned duration by the instructor. One group of students was solving the visual basic assignments and the other group of students was solving the assignments on Java. One group of students was given 16 assignments in VB and the other group was given 22 assignments in Java. The experiment is conducted in two phases in the first phase both the class solved half of the assignments in alone and in the second phase both the classes solved other half of the assignments in pair. Pairs were formed first by selecting top 30 students from each class to make the first partner and then for the second partner randomly selected students from the rest of the class to minimize the biasness.

4.4 Design

In this experiment single-factor block design is used. The subjects' capability to develop programs in pair was considered to be a blocking variable. To complete the design, the exercise was replicated on two phases, first set of exercise individually and in second phase in pair and hence they had the same influence on both the approaches. The various elements are described in the Table 1.

Table 1: Experiment element details

Elements	*Details*
Approaches used	Individual or solo and Pair
Variables measured	Effort, Efficiency
Blocking variable	Subject experience
Experimental Design	Single-factor block design
Development environment	Visual basic 6.0 and Java 1.6

5. Analysis and Finding

It was seen that most of the students participated very enthusiastically and were satisfied with their work, while using both solo and pair programming. The data collected from the experiment was analyzed using multivariate analysis or factor analysis and the factors were rotated through varimax procedure. Two factors (pair and solo) were extracted. The first factor pair had loading on 10 items and solo had loading on 9 items. Again when all items on pair factor were analyzed using principal component analysis and rotated through varimax procedure, two factors were extracted such as pair effort and pair efficiency. Both factors had loading on five items each. Then all the variables Pair, Solo, PEffort, SEffort, PEfficiency, and SEfficiency were calculated using the formula mentioned in section 4.2 After calculating all these variables the means were compared using paired-sampled t-test. The means of pair and solo, PEffort and SEffort, PEfficiency and SEfficiency were compared. All the results show significant difference in the performance of pair and solo programming. The results are shown in table 2 and results of the hypothesis are shown in table 3. The response of preferring pair programming is moderate i.e. 3.2 whereas response of solo is 2.5 slightly favoring solo. The paired sampled t-test shows the difference between the pair and solo is $t = 5.96$ at $p = .000$ and degree of freedom is 90.

Table 2: Effectiveness of pair in effort and efficiency

Item[1]	*N*	*M*	*SD*
Pair	91	3.20	0.54
Solo	91	2.58	0.68
T	$t_{90} = 5.96^{***}$		
PEFFRT	91	3.01	0.53
SEFFRT	91	2.63	0.76
	$t_{90} = 3.43^{**}$		
PEFCNY	91	3.50	0.73
SEFCNY	91	2.43	0.89
T	$t_{90} = 9.40^{***}$		

[1]PEFFRT = Effort in pair programming, SEFFRT = Effort in solo programming, PEFCNY = Efficiency in pair programming, SEFCNY = Efficiency in solo programming. T= t-Test value
N = No. of sample, *M* = Mean, *SD* = Standard deviation
* p <0.05, **p<0.01, ***p<0.001

Similarly the response of effort in pair programming is moderate i.e. 3.01 whereas response of solo is 2.63 slightly in solo. The paired sampled t-test shows the difference between the effort in pair and effort in solo is $t = 3.43$ at $p = .001$ and degree of freedom is 90. The result also shows the efficiency in pair programming is between moderate and high i.e. 3.5 whereas efficiency in solo is 2.43 slightly favoring solo. The paired sampled t-test shows the difference between the pair efficiency and solo efficiency is $t = 9.40$ at $p = .000$ and degree of freedom is 90.

The formula used for t-test:

$$t = \frac{\bar{d}}{s_d / \sqrt{n}}$$

where $\bar{d} = \sum_{i=1}^{n} d_i$ and $s_d^2 = \frac{1}{(n-1)} \sum_{i=1}^{n} \left(d_i - \bar{d}\right)^2$

Here d_i= difference of individual data, s_d= standard deviation and n=number of sample

Table 3: Results of hypothesis

Response variables	Hypothesis	t-values	Result
Pair & Solo	$\mu_{pair(Programing-technique)} = \mu_{solo(Programing-technique)}$	5.96^{***}	$\mu_{pair(Programing-technique)} > \mu_{solo(Programing-technique)}$
PEffort & SEffort	$\mu_{pair(effort)} = \mu_{solo(effort)}$	3.43^{**}	$\mu_{pair(effort)} \neq \mu_{solo(effort)}$
PEfficiency SEfficiency	$\mu_{pair(efficiency)} = \mu_{solo(efficiency)}$	9.40^{***}	$\mu_{pair(efficiency)} = \mu_{solo(efficiency)}$

Further, correlations between the factors were studied. The factors PEffort, PEfficiency, and SEffort, SEfficiency were correlated to see whether the impact of effort in pair and solo does have any significant effect on efficiency of code or not. The result shows effort does have a significant effect on pair than solo. Correlation coefficient of Karl Pearson denoted by r as given by equation

$$r(X,Y) = \frac{Cov(X,Y)}{\sigma_x \sigma_y} \qquad \text{(i)}$$

If (x_i, y_i); i=1, 2, 3, …,n is the bivariate distribution, then

$$Cov(x,y) = \mathrm{E}[\{X - \mathrm{E}(X)\}\{(Y - \mathrm{E}(Y)\}] = \frac{1}{n}\sum\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)$$

$$\sigma_X{}^2 = \mathrm{E}\{X - \mathrm{E}(X)\}^2 = \frac{1}{n}\sum\left(x_i - \bar{x}\right)^2$$

$$\sigma_Y{}^2 = \mathrm{E}\{Y - \mathrm{E}(Y)\}^2 = \frac{1}{n}\sum\left(y_i - \bar{y}\right)^2$$

The summation extending over i from 1 to n.

The relationship between PEffort and PEfficiency is significantly high i.e. value of $r = 0.618$ ** whereas the value of SEffort and SEfficiency is significant but low i.e. value of $r = 0.263$*. This indicate that effort put in pair results in 61.8% efficient result whereas the effort put in solo results in only 26% efficient results as per this data set. This shows the overwhelming acceptance of pair programming technique in coding as well as in analysis.

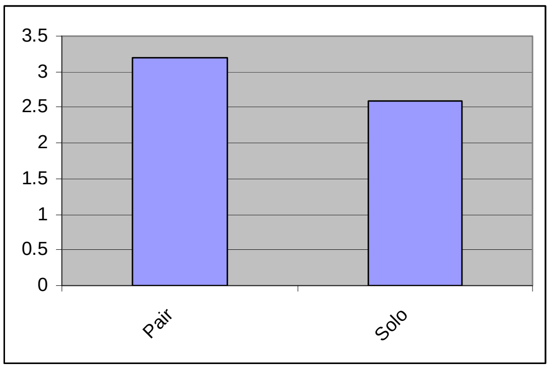

Figure 1: Mean of pair and solo

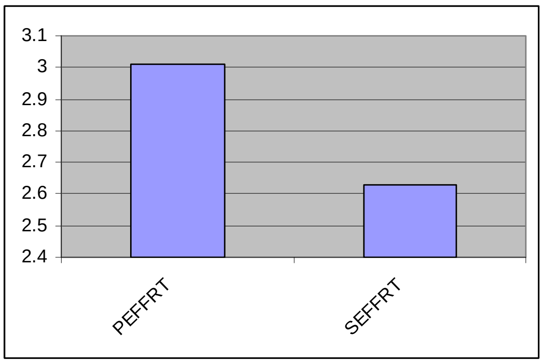

Figure 2: Mean of PEffort and SEffort

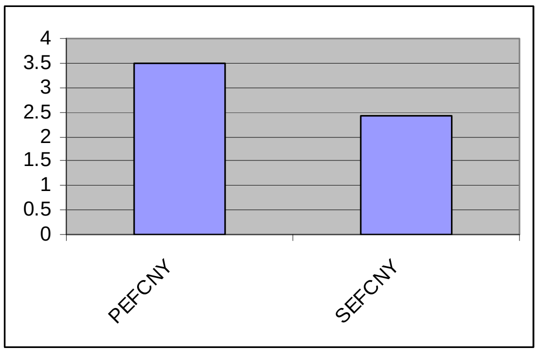

Figure 3: Mean of Pair Efficiency and Solo Efficiency

6. Summary

Pair programming has been compared with solo programming technique. The experiment started with a set of questionnaire with a Likert type 5 time scale and then factor analysis was done with principal component factor rotated through varimax procedure. The values generated were then compared with paired sample t-test. The result shows that pair programming is more efficient and effective in solving programming assignments.

7. Acknowledgment

I wish to extend my heartfelt thanks to Prof. Dr. Himadri Sinha of Xavier Institute of Social Service, Ranchi for his invaluable help in data analysis.

Reference

[1] Arisholm, E., Gallis, H., Dyba, T. , Sjoberg, Dag I.X., “Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise”, IEEE Transactions on Software Engineering, Vol. 33, No. 2, February 2007, pp. 65-85.

[2] Bhattacherjee, V., Neogi, M., Mahanti, R., “Are our Students Prepared for Testing based Software Development”, Paper presented and published in proc. for CSEET, Feb. 17-19, pp. 210-211, 2009a.

[3] Bhattacherjee, V., Neogi, M., Mahanti, R.”Software Development Patterns Of Students: An Experience”, National Journal of System and Information Technology, Vol2 (1), pp. 15-21, June 2009, 2009b.

[4] Bhattacherjee, V., Neogi, M.S., Mahanti, R., “VOSDM: Agile view for budding IT Professionals”, Accepted for presentation in International Conference on Business and IT: Contemporary Research and Development, IMT, Ghaziabad, 25-26, Feb. 2010.

[5] Bhattacherjee, V., Neogi, M., Mahanti, R., “Software Development Patterns in University Setting: A Case Study”, Proceedings of the National Seminar on Recent Advances on Information Technology, February 2007, ISM Dhanbad, pp 40-43.

[6] Bhattacherjee, V., Neogi. M., Mahanti, R., “Software Development Approach of Students: An Evaluation”, Proceedings of the National Conference on Methods and Models in Computing 2008 (NCM2C 2008), JNU, New Delhi, pp 23-30.

[7] Canfora, G., Cimitile, A, Garcia, F, Piattini, M, Visaggio, C. A., “Evaluating performances of pair designing in industry”, The Journal of Systems and Software, 80 (2007), pp.1317–1327

[8] Carver, J.C., Henderson, L., He, L., Hodges, J., Reese, D., Increased Retention Of Early Computer Science and Software Engineering Students using Pair Programming, 20th Conference on Software Engineering Education & Training (CSEET’07), 3-5 July 2007, pp.115-122, Digital Object Identifier 10.1109/CSEET.2007.29

[9] Dyba, T., Arisholm, E., Sjoberg, D.I.K., Hanny, J.E., Shull F, “Are Two Heads Better than One? On the Effectiveness of Pair programming”, Voice of Evidence, IEEE Software November/ December, 2007, pp. 12-15

[10] Kwak, Y.H., Stoddard, J., Project risk management: lessons learned from software development environment, Technovation, 24, 2004, pp. 915–920.

[11] Kyungsub, S. C., Fadi, P. D., Il Im, Exploring the underlying aspects of pair programming: The impact of personality, Information and Software Technology, 50, 2008, pp.1114–1126.

[12] Lui, K.M., Chan, K.C.C., Nosek, J.T., The Effect of Pairs in Program Design Tasks, IEEE Transactions On Software Engineering, Vol. 34, No. 2, March/April 2008, pp. 197-211.

[13] Melnik, G., Maurer F, “A Cross-Program Investigation of Students’ Perceptions of Agile Methods”, International Conference on Software Engineering archive Proceedings of the 27th international conference on Software engineering, ICSE ’05 May 15-21, 2005, pp 481-488.

[14] Muller, M.M., Do programmer pairs make mistakes than solo programmers?, The Journal of Systems and Software, In Press, available online at www.sciencedirect.com., Vol. 80, Issue 9, September 2007, pp. 1460-1471.

[15] Neogi, M. S., Mahanti, R., Bhattacherjee, V., Evolution of Software Process Models, Proceedings of the National Conference on Technological Advances and Emerging Societal Implications, NIT Rourkela, March 2007, pp 402-415,.

[16] Neogi, M., Bhattacherjee, V., Mahanti, R.,A Process Model for Software Development Amongst Students, International Journal of Recent Trends in Engineering Vol. 1, No. 2, May 2009, pp 69-74, 2009b.

[17] Neogi, M.S., Bhattacherjee, V., Mahanti, R. “Evaluating the Effectiveness of VOSDM – A Vision Oriented Approach”, ACM SIGSOFT Software Engineering Notes, Vol. 35, Issue 2, March issue 2010.

[18] Neogi, M.S., Bhattacherjee, V., Mahanti, R., An Evaluation of Student Preferences during Software Development, In Proc., National Seminar on Recent Advances Information Technology (RAIT ’09), ISM Dhanbad, Feb.6-7 2009, pp 239-245, 2009a.

[19] Succi, G., Pedrycz, W., Marchesi, M., Williams, L., Preliminary Analysis of the Effects of Pair Programming on Job Satisfaction,

Proceeding of the conference on Software Engineering: Evolution and Emerging Technologies, 2005, pp. 212-215.

[20] Succi, G., Stefanovic, M., Smith, M., Huntrods, R., Design of an Experiment for Quantitative Assessment of Pair Programming Practices, downloaded from http://agilealliance.com/system/article/file/1097/file.pdf, 2001, pp. 18-23.

[21] Toll III V. T., Lee R., Ahlswde T., "Evaluating the Usefulness of Pair Programming in a Classroom Setting ", 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), July 2007, pp 302-308.

[22] Vanhanen, J., Korpi, H., "Experiences of Using Pair Programming in an Agile Project", Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS), 2007, pp. 274–283.

[23] Vivekananda, K., Kuppuswami, S., "Students Attitude towards Pair Programming in Short Duration Laboratory Exercises". International Journal of Computer Science and System Analysis, July-December 2007 pp. 141-148.

[24] Williams, L., Layman, L., Osborne, J., Katira, N., "Examining the Compatibility of Student Pair Programmers", Proceedings of AGILE 2006 Conference (AGILE'06), 10 pp. – 420.

Appendix A

Variable	Factors Loadings				
	Unrotated		Rotated		Communality
	I	II	I	II	
1	.70	.13	.66	-.28	.50
2	.69	.18	.68	-.23	.51
3	.59	.21	.61	-.14	.40
4	.45	.49	.64	.16	.44
5	.59	.47	.75	7.247E-02	.56
6	.79	.13	.73	-.32	.64
7	-.54	.34	-.27	.58	.40
8	-.58	.55	-.18	.77	.63
9	-.72	.42	-.37	.75	.70
10	-.49	.43	-.18	.63	.43
11	-.38	.24	-.18	.41	.20
12	.59	.21	.61	-.13	.40
13	-.51	.36	-.22	.58	.39
14	.37	.20	.42	-3.223E-02	.18
15	.73	.16	.70	-.27	.56
16	-0.001	.56	.30	.47	.31
17	.31	.17	.34	-2.682E-02	.12
18	-0.05	.54	.25	.48	.30
19	0.04	.48	.30	.38	.23
Explained Variance	5.40	2.52	4.54	3.38	
Explained Variance (%)	28.41	13.28	23.91	17.78	

Appendix B

Variable	Factor Loadings on Pair Programming				
	Unrotated		Rotated		Communalities
	I	II	I	II	
1	.736	-.340	.780	.219	.657
2	.732	-.295	.748	.250	.623
3	.660	-.436	.785	.097	.626
4	.591	.066	.407	.434	.354
5	.702	.040	.508	.486	.495
6	.781	.242	.437	.691	.668
12	.629	.440	.193	.743	.589
14	.444	-.301	.534	.059	.288
15	.742	.213	.427	.643	.596
17	.334	.591	-.129	.666	.460
Explained Variance	4.22	1.14	2.92	2.43	
Explained Variance (%)	42.20	11.36	29.21	24.34	

Appendix C

Variable	Factor Loadings on Solo Programming				
	Unrotated		Rotated		Communalities
	I	II	I	II	
7	.658	-.331	.736	-.023	.543
8	.790	-.141	.776	.204	.644
9	.827	-.248	.855	.124	.746
10	.667	-.046	.624	.239	.447
11	.463	.356	.270	.518	.341
13	.668	-.190	.685	.109	.482
16	.344	.282	.194	.400	.198
18	.399	.652	.088	.760	.585
19	.231	.737	-.101	.765	.596
Explained Variance	3.18	1.40	2.86	1.72	
Explained Variance (%)	35.31	15.59	31.81	19.09	