

Impact of Pair Programming for Effective Software Development Process

¹P. Prabu and ² Dr. S. Duraisamy

¹ Assistant Professor, Department of MCA
prabukiwi@gmail.com

² Professor, HOD of Computer Applications,
Sri Krishna College of Engineering and Technology, Coimbatore

Abstract

Software development is a mentally complicated task. Different software development methodologies and quality assurance methods are used in order to attain high quality, reliable, and bug free software. eXtreme Programming (XP) is a software development discipline in the family of agile methodologies that contributes towards quality improvement using dozen practices. One important practice in XP is Pair Programming (PP). Pair or collaborative programming is a software development technique where two programmers develop software side by side at one computer. One person types in while the other reviews each line of code making it possible to create and continuously review what is being created. *Although there has been a large amount of research investigating pair programming in an industry setting, there has been little work done with pair programming in an academic setting.* The main objective of the research is to develop a model for improving the effectiveness of software development process using Pair Programming by conducting experiments among employees of valtech Software Company with considerably having various levels of programming skills. The study indicates that, after experiencing pair programming, most employees pointed out a stronger preference to work with another employee, believed that pairing made them more organized, and believed that pairing saved time on practical sessions. Our experimental results *found that despite some drawbacks, pair programming can be extremely beneficial.*

Keywords: Extreme Programming, Pair Programming, REAP.

I. Introduction

Each day, software applications grow larger and more complicated; Perhaps, then, it is best for the complexity of these applications to be tackled by two humans at a time. Much of the increased interest in PP is probably due to the introduction of extreme programming (XP) [2]. In pair work, both partners actually perform each activity together in collaborative manner, making it possible to create and continuously review what is being created.

Each collaborative pair sits shoulder-to-shoulder at one computer during all phases of development. One is the 'driver.' This engineer has control of the mouse, keyboard, or writing utensil and is actively analyzing the requirements and creating the design, code, or test. The other person is designated as 'navigator'. The engineer observes the work of the driver and identifying tactical and strategic defects in their work, thinking of alternatives, writing down "things-to-do," and looking up references. Explicitly, the pair periodically takes turns being the driver and the navigator. The main purpose of the practice is to overlap the production of software artifacts with their continuous review: the aim is to detect more defects and adjust implementation strategies just when the code is written.

Pair Programming is increasingly attracting the researchers' and practitioners' in both the industrial and academic fields [3]. In Pair Programming practice, the interaction among the programmers helps them in better understanding of the problem and transferring tacit knowledge [1].

Many of the previous work in pair programming were done by applying pair programming in the coding phase of software development. Pair Programming is not reserved solely to the coding phase [8], but it can be applied to other phases of the process such as analysis, and design

II. Background

In 1998 Nosek [5] conducted an experiment with five pairs and five individual professionals solving a challenging problem. They were asked to write a UNIX script that performed a database consistency check. The programmers were well versed in the Unix script but had not performed that kind of task before. The controlled experiment showed that pair programming shortened the elapsed time and produced better software quality than individual programming. This creates evidence that collaboration improves the problem-solving process and produces more efficient code

In an academic environment, the most cited study is probably that described in [9] in which 13 university students worked individually on a project and 28 chose to work in pairs. The findings showed that code produced by the pairs passed more automated tests over four different programming exercises. This resulted that pair programming in the software development yields better products in less time. The programmers feel happier, more confident.

However, a common feature of the existing studies is that the basic understanding of the effectiveness of pairs in the coding phase alone and they have not addressed in all the phase of software development. This paper focuses on developing a model for software development using Pair Programming and assessing

the effectiveness, variation in effectiveness between pairs of varying skills with respect to all phases of software development.

III. Software Development Model Using Pair Programming.

The software development process involves many stages namely requirement analysis, designing, implementation, testing and maintenance.

The development model represented in figure 1 contains all the stages and each and every stage is tuned by the practice of Pair Programming.

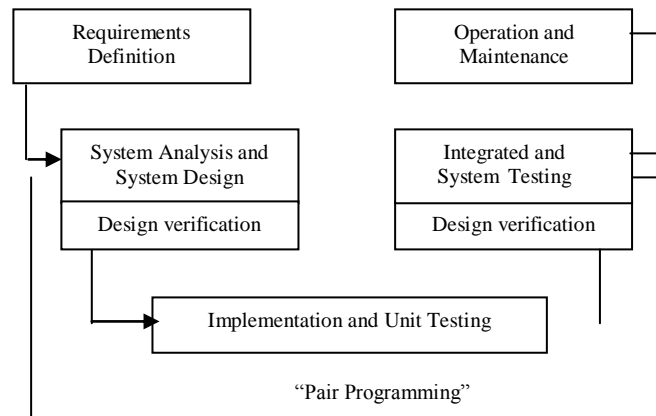


Figure 1: The Development Model

The analysis stage, actually a set model, is the first technical representation of a system. Pairing in the analysis phase is a constructive approach will result in an effective analysis of the system and comes out well defined models.

Software design requires dealing with many levels: implementation, database, business logic, presentation, deployment, interaction with other systems [7]. Pair working is applied to the design phase of the software development and results are given in terms of pair designing to improves performance.

Implementation needs a multi-layered, multidimensional model capable of supporting mental simulations and amount of knowledge required to its suitable organization [6]. One possible method of taming the complexity of software development may be to work collaboratively.

The figure 2, explains the practice of PP, in the implementation phase. Defects identified by the observer, is then there reviewed, the final outcome of process will be defect less or only has least number of defects.

The main advantage of the PP, the code done the driver is corrected by the observer in parallel manner.

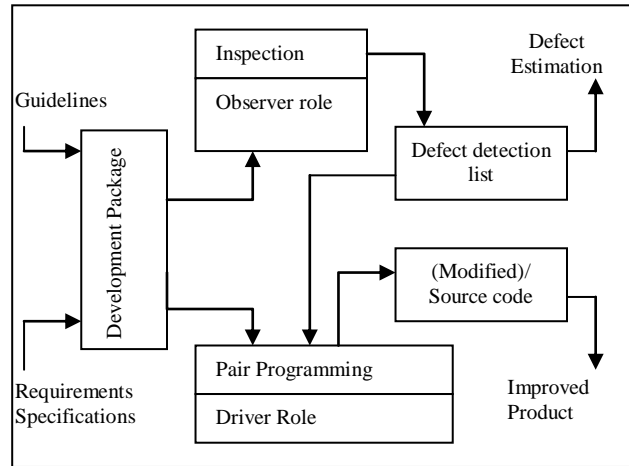


Figure 2: Pair Programming Approach

Testing is the most important way of assuring (or controlling) the quality of software. Good practices throughout the development process contribute to the quality of the final product [4]. Pair programming plays a major role in the testing phase by improving its effectiveness.

This research work presents the experimental set up, such a way that, the practice of pair working is extended to the testing phase and results are given in terms of performance, test case design and execution

IV. Experimental Setup

In this work the effectiveness of Pair Programming in software development process can be highlighted by conducting experiments between solo versus pair work at each software development phase.

We conducted an empirical analysis among 66 employees those are working as Software Engineer in Valtech. A skill test has been conducted for the employees to test their core conceptual knowledge, design oriented skill, programming skills and other skills for working in software projects. Based on the skill test result and the previous project performance, 36 employees were termed as Experts and 30 were termed as Novice.

Entire 66 employees were split into two groups comprising 20 and 46 employees. First group comprising 20 employees (10 experts & 10 novices) were made to work individually. The other group comprising 46 employees were made into 23 pairs. These 23 pairs were made in such a way that 8 Pairs are formed as Expert-Expert Pair, 10 pairs are formed as Expert- Novice Pair, and 5 pairs are formed as Novice-Novice Pair.

One programming problem was given to every pair and solo programmer to work upon to implement it. Analysis shown here is based on the code done by the pair and solo programmer and the effectiveness of pair programming were measured. Similar analysis is done in all phases.

V. Quantitative Results

The quantitative results analysis focuses on each and every stage of development done by the employees and facts obtained from the experimental study are given in the form parametric values taken for each phase of the software development process.

Parameter 1: Duration

Duration was defined as the total time taken to complete the analysis of the system. To end with a correct measurement, Time Log sheets were used to record the current time, total time (in minutes) for that task.

For the duration measure to be meaningful, we considered duration only employees with correct solutions. The table 1 gives the tabulated values of duration parameter.

In pair programming, duration can be measured in two ways: One is elapsed time to complete the task and the other is the total effort/time of the programmers completing the task.

Table 1: Tabulating Duration Parameter

PAIRS				
Subjects	Problem 1		Problem 2	
	Pair No.	Time taken (min)	Pair No	Time taken (min)
Expert - Expert	1	38	5	37
	2	43	6	41
	3	46	7	34
	4	35	8	45
Expert - Novice	9	38	14	53
	10	47	15	49
	11	48	16	55
	12	54	17	52
	13	49	18	57
Novice- Novice	19	56	22	65
	20	67	23	69
	21	63	-	-
INDIVIDUAL				
Expert	1	48	6	49
	2	51	7	52
	3	56	8	46
	4	45	9	54
	5	53	10	58
Novice	11	67	16	78
	12	72	17	67
	13	76	18	74
	14	78	19	69
	15	89	20	81

In this, we incorporate both important measurements of time in a single measurement, that is, the Relative Effort Afforded by Pairs (REAP). REAP is used to measure for non programming related tasks, for example analysis related tasks.

$$REAP = \frac{\widehat{\text{finish_time_of_pair}} \times 2 - \widehat{\text{finish_time_of_individual}}}{\widehat{\text{finish_time_of_individual}}} \times 100 \quad (1)$$

There are five cases for us to consider with REAP:

- REAP < 0,
- REAP = 0,
- REAP is between 0 and 100,
- REAP = 100, and
- REAP > 100.

When REAP is negative, the total time of pair programmers is less than the time of the individual programmer, that is, pairs are actually more efficient than a single pair programmer and it is less costly to use pair programmers than individual programmers. The table 2 gives the calculated REAP values for various pairs.

If REAP is zero, this is a break-even point, where the total time of pair programming is the same as individual programming, but pair programming halves the elapsed time required for individual programming.

When REAP is greater than zero but is less than 100 percent, pairs require more total man hours to complete the task but are faster than individual programmers, that is, the elapsed time to complete is less for pairs than for individual programmers.

If REAP is around 100 percent, the elapsed time for pair programmers is almost the same time as in the individual programmer; therefore, pair programming doubles the total man hours as compared to individual programming. When REAP is greater than 100 percent, then the elapsed time for pair programming is longer than the time for an individual programmer. The table 2 gives the REAP values for different level of pairs.

Table 2: Tabulation of Reap Values

Expert – Expert pair with Expert Individual			
Problem 1		Problem 2	
Pair No	REAP value	Pair No	REAP value
1	58.33	5	51.02
2	68.52	6	57.69
3	64.28	7	47.82
4	55.55	8	66.66
Expert – Novice pair with Novice Individual			
9	13.43	14	35.89
10	30.55	15	58.20
11	26.31	16	48.64
12	38.46	17	50.72
13	10.11	18	40.74
Novice – Novice pair with Novice Individual			
19	67.16	22	66.66
20	86.11	23	105.9
21	65.78	-	-

The REAP values obtained for this pair is greater than zero but less than hundred. This implies pair requires more man hours but the pairers are faster than Individuals.

The results show that many of the pair programmers were done faster than that of the Individual programmers. Only few of the pairs results does not show significant difference between pair programmers and Individual programmers

The results show that, some of the pairs are really faster than individual. Similar manner duration and effort analysis is done for all the phases of software development process and found the results were favor to pair work

Parameter 2: Correctness of the Solution

Correctness of the solution in the analysis phase can be defined in the following way
a) All the needs of the customer as transferred to requirements of the software. b) Use cases identifications are done correctly. The measurement of the correctness of the solution is done by means of Binary functional correctness score. The binary score is with value 1, when both the tasks t1 and t2 are done correctly. The binary score is 0, if any one of the tasks t1 and t2 contains errors. the pair and solo group for the given. The correctness of the solution parameter can be taken in all phases and the results are given. The tabulated values of correctness of the solution are given table 3.

The mean values are taken according to subject for the result analysis in the comparative way. The table 4 shows the mean values.

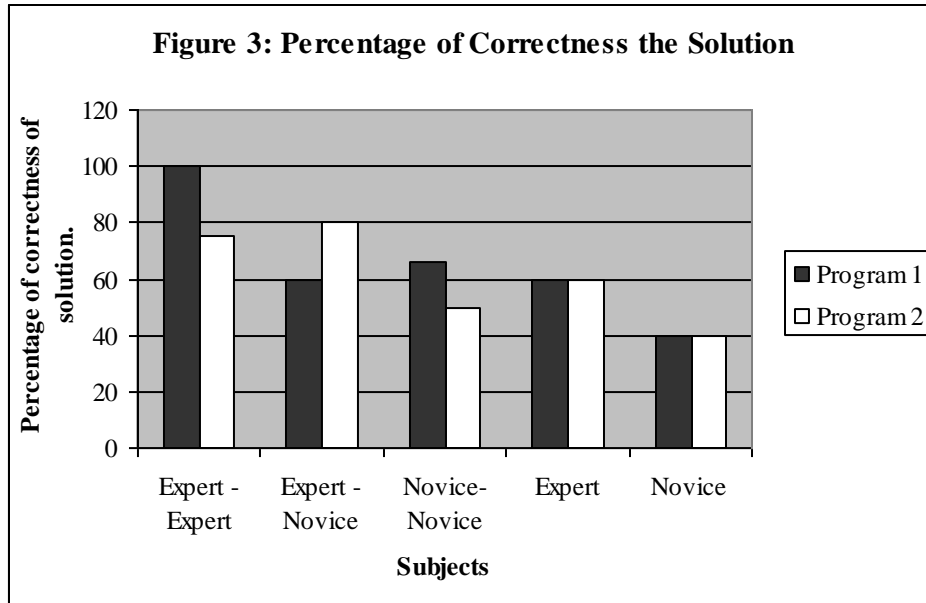
Table 3: Correctness of the Solution

PAIR				
Subjects	Problem 1		Problem 2	
	Pair No.	Binary score	Pair No	Binary score
Expert – Expert	1	1	5	1
	2	1	6	0
	3	1	7	1
	4	1	8	1
Expert – Novice	9	1	14	1
	10	0	15	1
	11	1	16	1
	12	0	17	0
	13	1	18	1
Novice- Novice	19	1	22	1
	20	1	23	0
	21	0	-	-
INDIVIDUAL				
Expert	1	1	6	1
	2	1	7	1
	3	0	8	1
	4	1	9	0
	5	0	10	0
Novice	11	1	16	0
	12	1	17	0
	13	0	18	1
	14	0	19	1
	15	0	20	0

Table 4: Mean Values of Correctness of the Solution

Subject	Program 1	Program 2
	Mean value	Mean value
Expert - Expert	1	0.75
Expert - Novice	0.6	0.8
Novice - Novice	0.66	0.5
Expert	0.6	0.6
Novice	0.4	0.4

The graphical representation of the percentage of correctness of the solution of solo versus pair programmers are figured in figure.3. The graphs show pair programmers have more correctness in their solutions.



Parameter 2: Project size

Program size is given in terms of Number of physical lines of code. Comparison of average Loc of different level of complex programmers between the pair and solo group are given.

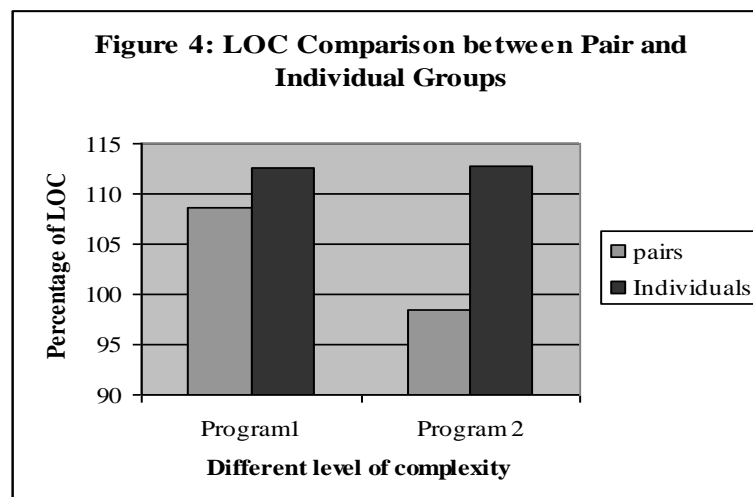
The Table 5 gives the values of the program size of the program 1 and program 2 done by the pair programmers and individual programmers groups.

The Figure 4 depicts the graph of comparison of percentage of Line of code of two different programs done by the two different groups of programmers.

The result shows that program developed by pair group has less number of lines of code when compared with program developed by individual group.

Table 5: Lines of Code

PAIRS				
Subjects	Program 1		Program 2	
	Pair No.	LOC	Pair No.	LOC
Expert - Expert	1	85	5	62
	2	76	6	73
	3	89	7	54
	4	94	8	56
Expert – Novice	9	112	14	123
	10	125	15	131
	11	123	16	97
	12	98	17	92
	13	110	18	118
Novice-Novice	19	142	22	148
	20	138	23	129
	21	112	-	-
INDIVIDUAL				
Expert	1	110	5	112
	2	87	6	82
	3	54	7	67
	4	63	8	78
	5	76	10	96
Novice	11	145	16	127
	12	132	17	156
	13	138	18	143
	14	167	19	129
	15	154	20	138



Parameter 4: Program quality

Program quality was reported as number for test cases passed, number of correct solutions of programming tasks (functionality) measured in terms of code metrics, readability of the code measured in terms of comment line percentage, well defined variable name's percentage.

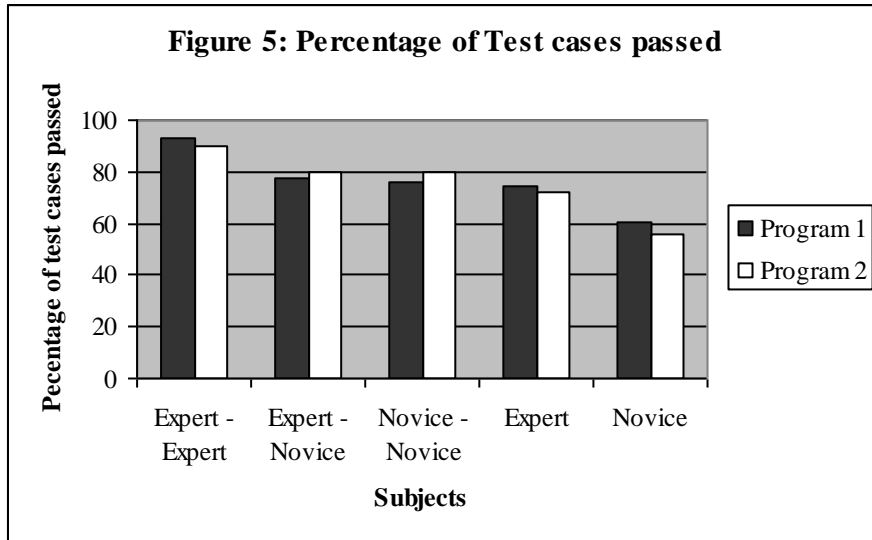
The values of number of test cases passed by the two different programs developed by different groups are taken to show that pair group produces high quality code.

Table 6 depicts the values of number test cases passed by two different programs by the pair and individual group. The total number of test cases defined for program 1 is 7 and for program 2 are 5.

Table 6: Numbers of Test Cases Passed

PAIRS				
Subjects	Program 1		Program 2	
	Pair No.	No of test cases passed	Pair No.	No of test cases passed
Expert – Expert	1	7	5	5
	2	6	6	4
	3	7	7	5
	4	6	8	4
Expert – Novice	9	7	14	5
	10	6	15	5
	11	5	16	4
	12	5	17	3
Novice – Novice	13	4	18	3
	19	6	22	4
	20	5	23	3
	21	5	-	-
INDIVIDUAL				
Expert	1	6	6	3
	2	6	7	4
	3	5	8	3
	4	5	9	3
	5	4	10	5
Novice	11	5	16	4
	12	6	17	4
	13	4	18	2
	14	3	19	3
	15	3	20	2

The graphical representation of percentage of test cases passed by the two different programs done by different programming groups is shown in the figure 5.



The graphical representation shows that the percentage of test cases passed by the program developed by pair group is greater than the test cases passed by program developed by individual group.

Other aspects of developing the software in all the phases of development also taken in to consideration and those factors are also have positive impact done by the pairs.

VI. Formal Tests of Hypotheses

In this research, the paired sample t- test(Easton *et al.*, 1997) has been used as the statistical technique for validating the experimental results. Generally the paired sample t-test is used to determine whether there is a significant difference between the average values of the same set of quantifications made under two different conditions. The usual null hypothesis is that the difference in the mean values is zero. This null hypothesis may be tested against any of the appropriate alternative hypotheses, depending on the question posed. The paired sample t-test is a more powerful alternative for the two sample t-test.

By adopting the principle of the paired-sample t-test, here it is conducted to establish the effectiveness of the pair programming over individual. In this research, the paired sample t-test is being carried out in three steps as described below:

Step-1: Definition of Hypotheses:

The null and alternate hypotheses are defined such that to factually explore the differences between the proposed and existing methodologies as given below:

- i). *Null Hypotheses (H0)* : There is no difference between the pair programming and individual programming.
- ii). *Alternative Hypotheses (H1)*: There exists difference between the pair programming and individual programming.

Step-2: Calculation of t-statistic for the given data set:

The theoretical estimate of the t-value can be found using the following Eqn. (7.10).

$$t_{calc} = \frac{M_x - M_y}{\sqrt{\frac{S_x^2}{n_x} + \frac{S_y^2}{n_y}}} \quad (2)$$

where,

$$S^2 = \frac{\sum(x-M)^2}{n-1}.$$

M_x – Mean of sample x, here x refers to the pair programming approach.

M_y – Mean of sample y, here y refers to the Individual approach.

S_x^2 – Variance of sample x.

S_y^2 – Variance of sample y.

n_x – Sample size of x.

n_y – Sample size of y.

df = $n_x - 1$ or $n_y - 1$, when $n_x = n_y$ and $S_x^2 \neq S_y^2$.

Step-3: Comparison between the tabulated and calculated values:

The term t_{calc} refers to the estimated value of t , whereas t_{tab} refers to the tabulated t -value for the appropriate significance level and degree of freedom. If $t_{calc} > t_{tab}$, null hypothesis H_0 is rejected and thereby the alternate hypothesis H_1 is accepted. Otherwise, the H_0 should be accepted by rejecting the H_1 .

VII. Statistical Results

The paired sample t-test was conducted over the two different result sets of Expert and Novice for the estimate Time taken (Tt) and are tabulated in Table 7. The findings of the statistical analyses from Table 7 are reported as follows:

- The t-test results of Tt(Time taken) shows that there is a significant difference between the scores for Expert (M=50.6, SD=4.277) and Novice (M=76.4, SD=8.20). Since the absolute value of t_{calc} (-7.17) is greater than t critical two-tail (2.77), Null Hypothesis (H_0) is rejected. Moreover since the probability $P(T \leq t)$ two-tail (0.002) is less than α (0.05), Null Hypothesis is strongly rejected and hence the Alternate Hypothesis (H_1) is accepted at the confidence level of 95%.

Table 7: Measures of Time Taken (Tt) for Expert vs Novice

S.No	Problem 1		Problem 2	
	Expert	Novice	Expert	Novice
1	48	67	49	57
2	51	72	52	59
3	56	76	46	51
4	45	78	54	69
5	53	89	58	72
Paired Sample T-Test Result				
Mean (M)	50.6	76.4	51.8	61.6
Std. Deviation (SD)	4.27785	8.203658	4.604346	8.70632
Variance	18.3	67.3	21.2	75.8
Observation	5		5	
Degree of Freedom	4		4	
t_{calc}	-7.1722029		-4.937168674	
P(T<=t) one tail	0.001000513		0.003916303	
T Critical one tail	2.131846786		2.131846786	
P(T<=t) one tail	0.002001027		0.007832605	
T Critical two tail (t_{tab})	2.776445105		2.776445105	
Confidence Level (α)	95% (0.05)		95% (0.05)	

Table 8: Measures of Duration taken by different pairs with Statistical Result- Problem 1

Completion Time

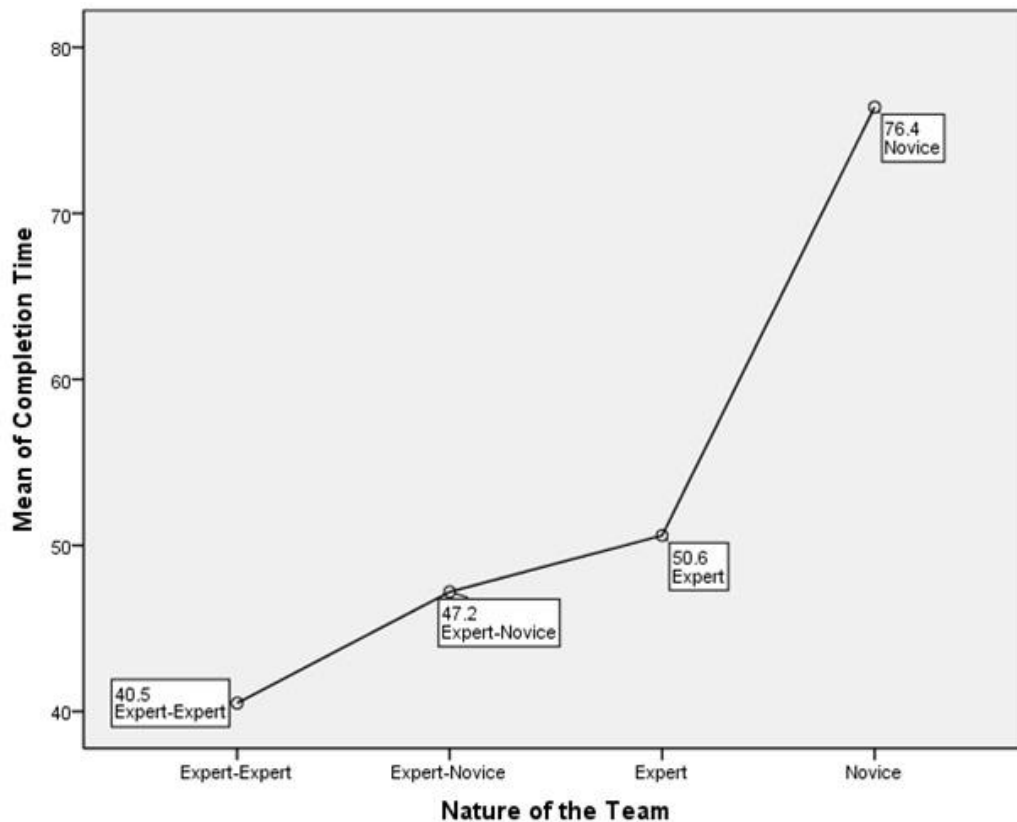
	N	Mean	SD	SE	95% Confidence Interval for Mean		Min	Max
					LB	UB		
Expert-Expert	5	41.20	4.550	2.035	35.55	46.85	35	46
Expert-Novice	5	47.20	5.805	2.596	39.99	54.41	38	54
Novice-Novice	5	68.00	10.050	4.494	55.52	80.48	56	83
Total	15	52.13	13.627	3.518	44.59	59.68	35	83

Completion Time

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1978.133	2	989.067	19.09	.000
Within Groups	621.600	12	51.800		
Total	2599.733	14			

If $p < 0.5$, so we reject the null hypothesis. The means of the three populations are not all equal. At least one of the means is different. So Expert-Expert pair is better than Expert individual. In the test case number of test pass of Expert-Novice is higher than the Expert individual. If $F > F_{crit}$, we reject the null hypothesis. This is the case, $19.09395 > 3.885294$. Therefore, we reject the null hypothesis.

VIII. Statistical graph between Pair and Individual



IX. Conclusion

In this work we proposed a model for software development using pair programming suitable for academic environment. We conducted an experiment using the employees of valtech Software Company in order to find the effectiveness of pair programming. The results shows that Pair programming is more effective with respect to fastness in completion, higher quality, program size, defect identification speed and defect removal rate, number of reworks done etc. From the results of the experiment we can conclude that pair programming is very effective in the entire software development process and can be incorporated in the industry environment

References

- [1] E. Arisholm et al., "Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise," *IEEE Transactions on Software Engineering.*, vol. 33, no. 2, 2007, pp. 65–86
- [2] Beck, K., *Extreme Programming Explained: Embrace Change*. 2000, Reading, Mass: Addison-Wesley.
- [3] K. Beck, "Embrace Change with Extreme Programming," *Computer*, vol. 32, no. 10, pp. 70-77, Oct. 1999.
- [4] Glenford J. Myers, "The Art of Software Testing", Second Edition
- [5] John T. Nosek, The Case for Collaborative Programming, *Communications of the ACM*. March 1998/Vol. 41, No. 3.
- [6] Perry, D., N. Staudenmayer, and L. Votta, *Understanding software development: Processes, organisations and technologies*. IEEE software, 1994. 11(4): p. 36-45
- [7] Pressman, "Software Engineering, A Practitioner's Approach", Sixth Edition.
- [8] Williams I. and Kessler R.R. *Pair Programming Illuminated*: Pearson education Inc., 2003
- [9] Williams, L., et al., *Strengthening the Case for Pair Programming*. IEEE Software, 2000. 17(4): p. 19-25.
- [10] Laura Plonka, Helan Sharp, Janet Van der Linden and Yvonne Ditrich "Knowledge Transfer in pair programming: An In-depth Analysis" *ELSEVIER*, 2014 73(2015) 66-78
- [11] Brian Hanks "Empirical Evaluation of Distributed pair programming" *ELSEVIER*, 2007 66(2008)530-544
- [12] Stapel, K., E. Knauss, K. Schneider, and M. Becker, *Towards Understanding Communication Structure in Pair Programming*, in *Agile Processes in Software Engineering and Extreme Programming*, A. Sillitti, et al., Editors. 2010, Springer Berlin Heidelberg. p. 117-131.
- [13] Williams, L.A. and R.R. Kessler, *All I really need to know about pair programming I learned in kindergarten*. *Communications of the ACM*, 2000. 43(5): p. 108-114.
- [14] Zarb, M., J. Hughes, and J. Richards, *Analysing Communication Trends in pair programming using Grounded Theory*, in *Proceedings of the 26th BCS Conference on Human-Computer Interaction*. 2012, British Computer Society: Birmingham, United Kingdom.
- [15] Zarb, M., J. Hughes, and J. Richards, *Industry-inspired guidelines Improve students pair programming communication*, in *Proceedings of the 18th Conference on Innovation and technology in computer science education*. 2013, ACM: Canterbury, England, UK. p. 135-140.
- [16] Hanks, B., S. Fitzgerald, R. McCauley, L. Murphy, and C. Zander, *Pair programming in education: a literature review*. *Computer Science Education*, 2011. 21(2): p. 135-173.
- [17] McDowell, C., B. Hanks, and L. Werner, *Experimenting with pair programming in the classroom*. *SIGCSE Bull.*, 2003. 35(3): p. 60-64.

- [18] Srikanth, H., L. Williams, E. Wiebe, C. Miller, and S. Balik, *On Pair Rotation in Computer science Course in Proceedings of the 17th Conference on Software Engineering Education and Training*. 2004, IEEE Computer Society. p.144-149.



Mr. Prabu is currently working as an Assistant Professor in the Department of Computer Application at Sri Krishna College of Engineering & Technology, Coimbatore, Tamil Nadu, India. He has more than 7 years of teaching and industrial experience. His research interests include Software Engineering, Webservice, Open Source Tools and Mobile Application.



Dr.S. Duraisamy has over 16 years of experience in teaching and is currently heading the Department of Computer Applications at Sri Krishna College of Engineering and Technology, Coimbatore, India. He received his Doctorate in Computer Science and Engineering from Alagappa University, Karaikudi, India. He has published many papers in various reputed international journals and guiding dozens of research scholars. His areas of interest include software engineering, object oriented concepts and wireless sensor networks.