ANALYZING STUDENT LEARNING OUTCOMES IN PROGRAMMING COURSE USING INDIVIDUAL STUDY VS. PAIR PROGRAMMING

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Keith Stefan Abeyratne

In Partial Fulfillment of Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

August 2014

Fargo, North Dakota

North Dakota State University

Graduate School

Title

ANALYZING STUDENT LEARNING OUTCOMES IN PROGRAMMING COURSE USING INDIVIDUAL STUDY VS. PAIR PROGRAMMING

By

Keith Stefan Abeyratne

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Gursimran Walia

Chair

Dr. Saeed Salem

Dr. Janet Knodel

Approved:

8/25/2014	Dr. Brian Slator
Date	Department Chair

ABSTRACT

Pair programming has been common practice in the programming industry during last three decades, but only recently did it start to draw the attention as a teaching strategy. This paper investigates whether we should introduce pair programming at the beginning of the semester, instead later in the semester. To perform this investigation, we performed a control group empirical study wherein pair programming was used in the first half of the semester (in one section of introductory CS course). The control group (the other section of the same course) introduced pair programming in the second half of the semester.

This study supported the implementation of specific assessment strategies to assess individual programming abilities during pair programming situations. Results found that students perceive pair programming as being beneficial and all of the subjects who used pair programming indicated that they would prefer using it again as opposed to working individually.

ACKNOWLEDGEMENTS

This research could not have been accomplished with only my contribution, but there were many individuals who genuinely helped me in various ways. First and foremost, I would like to thank my advisor Dr. Gursimran Walia for accepting me as a student, and providing me freedom to learn. His guidance, encouragement, patience, thoughtfulness, and genuine support for my research, academic achievements, as well as completing my thesis were enormous. I would also extend my gratitude to Dr. Saeed Salem and Dr. Janet Knodel for accepting to me on my committee in such short notice. This wouldn’t be possible without your help and quick response.

I would like to give my gratitude to my colleagues at NSDU Computer Science department for their support, friendship and collegiality, especially staff at the NDSU Graduate School and computer science department, without your help and guidance this wouldn’t have been a reality.

Last but not least my heartiest gratitude goes to my parents and my family, for giving me their blessings, strength, love, comfort, guidance, support, courage and the best education. I owe them for giving me their best throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Pair programming (PP), by definition, is a programming technique in which two programmers work together at one computer on the same task [1]. The term “pair programming” was first used in 1999 as one of the core practices in the Extreme Programming (XP) software development methodology in industry. As defined by Williams et al. [2], pair programming refers to a practice in which two programmers sitting side by side using only one computer to work collaboratively on the design, algorithm, code or test. The person typing is called a driver, and the other partner is called a navigator. Both partners have their own; responsibilities; the driver is in charge of producing the code. The navigator’s tasks are more strategic, such as looking for errors, thinking about the overall structure of the code, finding information when necessary, and being an ever-ready brainstorming partner to the driver. This arrangement leaves the driver free to work on the tactical aspect of the program.

Challenges of creating rigorous, syntactically correct code without worrying about the big picture, gives the navigator the opportunity to consider strategic issues without being distracted by the details of coding. Together, the driver and navigator create higher-quality work more quickly than either could produce on their own. Pair programming is one of the key practices in Extreme Programming (XP) [3]. It was incorporated in XP, because it is argued to increase project members’ productivity and satisfaction while improving communication and software quality [3]. Since then, pair programming has become one of the most researched topics in the realm of agile software development techniques [4].

The practice of pair programming has been widely implemented in the industry as well as in educational settings (Domino et al [6], Chong et al. [7]). A vast amount of research on pair programming has been conducted to observe the benefits of the technique and to understand how the practice can improve students' learning outcomes. The stated benefits of pair programming were identified as follows:

- Improvement in students' academic performance such as in final and midterm exams, quizzes, programming assignments and overall course grades [13, 14, 15, 18, 16, 17].
- Improvement in programming productivity in terms of the time spent on coding and quality of the software produced [24, 02, 24, 25, 08].
- Increase in students' retention rate and course completion rate [13, 14, 18].
- Increased students' confidence level and enjoyment in learning programming [02, 13, 14, 16, 17].
- Reduced staff workload [22, 23].
- Increased efficiency in helping female students to work in programming tasks [19, 20, 21].

Recently, many researchers have explored the suitability of pair programming to conduct the programming laboratory classes in educational institutions. The pair programming can be considered as one form of collaborative learning. In collaborative learning, small groups of students associate with each other where each member contributes his/her personal experience, information, perspective, insight, skills and attitudes, which can help, improve the learning efficiency of others [12] (Klemm, 1994). When this kind of collaborative learning is adopted to do programming assignments, generally the students divide the work among them and complete it individually with little or no help from other students of the group.

1.1. Problem Statement

In literature, many benefits of pair programming have been proposed, such as increased productivity, improved code quality, and confidence, to name a few. On the other hand, pair programming has also received criticism over increasing efforts, expenditure and overall personnel costs, and bringing out conflicts and personality clashes among developers. However, the scientific empirical evidence behind these claims is currently scattered and unorganized, and thus it is difficult to draw conclusions whether the pair programming indeed claims to be beneficial for student learning or not. Researchers have also investigated the effect of the factors (e.g., student's skill levels) on the pair performance of students.

In fact, Hanks [5] points out regarding the quality improvement claims that "There does not appear to be any empirical evidence that the programs [produced by pair programming] are better in terms of design, readability, maintainability, or other internal quality attributes." As a consequence, the industry has been rightfully hesitant in adopting the pair programming practice. On the contrary a study by John Nosek [33] examined how pair programming impacted the amount of time it took to complete a programming task. His results found that pairs took about 70% of the time to complete a task as individuals. There are many studies that conclude pair programming helps create better quality code, and improve student learning.

A recent study held at NDSU by Radermacher et.al [26,27,28,29,30] indicates some valid points, which may be overlooked in other studies. Radermacher's study [30] provided increased support that the pair programming is more effective in improving student's learning of programming concepts compared to individual learning Radermacher found that overall assignment scores improved for students participating in pair programming.

Pair programming has proved its usefulness in teaching and learning programming skills. It also has received many criticism; therefore, the data we have is widely scattered. Furthermore, there are very few studies which indicate when pair programming should be introduced in introductory computer science courses to maximize its benefits. The main objective is to understand when pair programming should be introduced in courses for the students. Our goal is to use pair programming to improve student's learning capacity.

1.2. MOTIVATION AND RESEARCH GOALS

In recent years, the growth of extreme programming (XP) has brought considerable attention to collaborative programming. Pair programming is a collaborative approach that makes working in pairs rather than individually the primary work style for code development. Because pair programming is a radically different approach than many developers are used to, it can be hard to predict the effects when a team switches to pair programming. Despite the advantages proposed for pair programming, many still suspect the overall usefulness and benefit over traditional solo programming. One of the most questioned aspects lies on the feasibility of achieving superliner speedup as compared to the legacy pattern [02].From a simple management's viewpoint, there is no reason to pair up developers if they cannot do things twice faster. In addition to productivity, people also doubt if the improved software quality deserves hiring twice many programmers [02]. Although pair programming is claimed to cost an insignificant 15% more effort yet achieve a higher quality than solitary programming on the same task [02], the paradigm is still questioned if solo programming plus an additional review phase, which might be cheaper and equally effective, will achieve the same goal. The purpose of this study was to investigate the effects of pair programming on student performance based on the time it was introduced, and subsequent pursuit of computer science related degrees among college students.

2. BACKGROUND & RELATED WORK

The pattern of pair programming in which two individuals develop one software module together has attracted researcher and practitioner's interest for almost 15 years. People following the pair programming protocol sit in front of one screen, use a set of keyboard and mouse to collaboratively solve a programming task [02]. While one developer is modifying the source code, another is required to perform continuous code-review. This pattern was claimed by it advocators to yield earlier release and higher software quality [02], and was included as one of the rules of Extreme Programming (XP) [11], a popular software development methodology used widely in the software industry.

Studies conducted on pair programming identified some of its advantages for the teaching learning situation (Tomayko, 2002; Williams & Kessler, 2001; Williams & Upchurch, 2001). For example, it has been found that when programmers work in pairs, fewer errors are made than in individual programming situations (Tomayko, 2002), resulting in better programming performance, increased confidence, and decreased frustration levels of the programmers (VanDeGrift,2004). A possible explanation for these findings could be that pair members help each other to solve the problem and complete the programming task together. Thus, there seems to be some agreement among researchers that pair programming could be a promising teaching strategy for teaching programming skills (VanDeGrift, 2004).

Despite the possible benefits of pair programming as a teaching strategy, some of its limitations for student assessment have been documented as well. One such frequently encountered limitation is that some students may receive undeserved credit for the successful completion of a program (McDowell, Hanks, & Werner, 2003). The assumption is that it is difficult to assess students' individual programming abilities reliably in pair programming situations. The first author of this article experienced a similar problem when implementing pair programming as a teaching strategy during a second year Information Technology (IT) course for student teachers. The students achieved high marks for their pair assignments, but significantly lower marks for their individual assignments. This begs the question whether the results of pair programming assignments are reliable indicators of students' individual programming abilities. Naude and Hörne (2006) even classify undeserved credit under the umbrella of 'cheating,' and although cheating is a strong word, the researchers emphasize the seriousness of the situation.

Several researchers highlight the problem of assigning group marks to individual students (Parsons 2004). It is necessary to determine the contribution of each individual member, and according to Parson (2004), this is not an easy task. Some students could contribute little or nothing at all, and if there is no assessment of students' individual contributions, marks awarded could be an inaccurate reflection of a student's abilities (Cheng & Warren, 2000). According to Parsons (2004), a fair mark allocated to a given student should reflect that individual's effort and abilities. This statement is supported by the research of Verhaart, Hagen, and Giles (2005), who also wished to determine whether students' marks in group assessments correlated with their marks in individual assessments. They proposed two different assessment methods as best practice for assessing an individual's performance in group work. In the base mark adjusted method, they give individual tests, as well as self- and peer assessments after every group work session. In the task splitting method, they split the group work task and require some work done individually another work done in groups. A specific weighting is allocated to the different tasks and students need to include a peer review to adjust the group contribution. Verhaart et al. (2005)

concluded that both these methods are “valid assessment forms, producing marks which seem to reflect the students’ typical level of achievement.”

Although it cannot be clearly concluded from previous research whether pair programming reduces the development effort of students it improves the quality of the artifacts. Additional research may assist practitioners to know when and how to make pair programming more effective. Research has just begun to understand the impact of pairing individuals with different levels of expertise [17, 3]. Combining pair programming with solo programming are also being researched [18, 15]. Furthermore, there are evidence showing that pair programming might help teaching activities in Computer Science programs, for example, improving student confidence and course performance.

2.1. NDSU Studies

At North Dakota State University, pair programming was introduced in the introductory Computer Science programming courses (CS 160 and CS 161) beginning 2010.[26, 27, 28, 29 and 30] A series of empirical studies (SIGCSE’2011, CSEE&T’2011, SIGCSE’2012, ICER’2012, FECS’2014) showed that Pair Programming had a significant positive impact on the students’ acquisition of programming concepts and learning outcomes. Radermacher and Walia et al. 2012, studied multiple aspects of pair programming and have conducted several empirical studies North Dakota State University over the past two years. During this time, researchers received valuable feedback from course instructors about the effects of implementing pair programming in their introductory computer science courses. However these instructors also expressed concerns about the use of pair programming in their courses. These include being able to ensure equal participation from pair members and not being able to assess individual learning outcomes effectively

The following section briefly talks about the NDSU studies and the problems and concerns with the use of Pair Programming that led us to the current study (i.e., *impact on the learning outcomes by introducing Pair programming in the beginning half vs. later half in the course*) being reported in this paper.

NDSU Studies: A series of studies related to pair programming were conducted at NDSU [26, 27, 28, 29, 30]. Radermacher et al. 2012 , reports on the results of two different studies conducted during the spring 2010 semester [31]. Subjects in the first study were 35 students enrolled in one section of the CS1 course and the second study included 39 students enrolled in two sections of the CS2 course. Subjects in the CS1 course were split into two groups, one which used pair programming and one that did not; whereas the subjects in the CS2 course were paired based on declared major. Researchers reported that subjects from both the CS1 and CS2 courses indicated that they felt pair programming improved their understanding of programming concepts. Another major result indicated that pairing a computer science (CS) student with a non-computer science (nonCS) student produced less compatible pairs as compared to CS-CS pairs and nonCS-nonCS pairs.

Radermacher et al. 2012, reported another empirical study that investigated the effects of pair programming on student-instructor interactions during programming laboratory sessions. Subjects in this study were 44 students enrolled in one section of the CS1 course and 53 students enrolled two sections of the CS2 course during the fall 2010 semester. Subjects in the CS1 course alternated between using pair programming and working individually during lab sessions, whereas subjects in the CS2 course only used pair programming. Researchers monitored these lab sessions, marking the number of questions asked, how long it took before an instructor could address the subject’s question, and how long the instructor spent interacting with the subject.

Results of the study indicate that when pair programming is used, students spend less time waiting for assistance from an instructor and spend more time interacting with the instructor, likely due to a decrease in questions related to syntax errors or other minor problems.

Another experiment at NDSU investigated the effects of pairing subjects based on their mental model consistency levels (ranging from highly consistent to highly inconsistent) at the beginning of the semester to evaluate changes in the students' mental model consistency and their programming performance [27, 28]. The evidence suggest that such a pairing strategy can be an effective way if previous performance data is not available and that certain mental-model-based pairing arrangements (and not all) are more effective in motivating students towards greater consistency and resulted in better performance on exams.

2.2. PROBLEMS WHEN USING PAIR PROGRAMMING AT NDSU

Individual Assessment: Difficulties assessing individual learning and ensuring that both partners are benefiting from the use of pair programming is a common issue with pair programming. One of the CS1 instructors at our university also expressed concerns that several students, who he felt would not have normally been able to pass the class, had been able to pass the class because their partner was able to help carry them. Based on eight semesters of historic data for this instructor's class, there was a large increase in the number of students who were able to pass the course when pair programming was used and when students worked individually.

Results for the drop-rate of the course during the fall 2010 semester was about 8% when students used pair programming compared to the historical drop-rate of approximately 18%. Bevan, et al 2005. Had also reported that students were willing to submit assignments that only one of the students had completed. Williams, et al 2002. Also expressed similar concerns when they discovered instances of students who performed well on pair programming exercises, but scored poorly on exams, suggesting that one partner may have been completing most or all of the work on the programming exercises, this phenomenon also was observed by our own instructors. Other instructors indicated that this may have been an issue as a large number of laboratory assignments that required substantial out-of-class work in order to complete in summary there was no easy way to ensure that both partners had contributed equally.

How much Pair Programming: All three of the instructors with whom we worked have expressed some concern that using pair programming for every assignment may not be as effective as only using it for only some assignments. One of the CS1 instructors felt that most new students did not have any programming experience and that until they gained some knowledge they wouldn't be able to effectively pair. One instructor noted instances where one member of a pair would actually be providing the other with incorrect information.

Another instructor felt that some assignments should be completed individually in order to provide the opportunity for students to show that they have mastered the ability to program without the need of a partner. The instructor stated that occasionally one partner would be absent from a lab and that the remaining student seemed to struggle, even though that student had done well on previous lab exercises.

Previous research also has indicated that some students feel that it is important to work individually. There is no doubt that pair programming is beneficial to student learning and enjoyment and has benefits for instructors as well. It has already been suggested that pair work should not constitute the majority of a student's grade, but has not been determined how much pair activity should be conducted for efficiency of learning.

In introductory computer science courses where a majority of subjects have little or no experience with programming, our instructors felt that they should spend some initial time at the beginning of the course working individually in order to acquire some programming knowledge before working with a partner. Such an arrangement has several benefits. First, it provides performance data that can be used to pair based on ability, as most researchers agree that this is the most desirable pairing strategy. Secondly, it allows some time for students who do not intend to complete the course to drop before pairs are created, minimizing the need to re-pair students. Additionally, initial programming assignments are more likely be trivial or straightforward and may not benefit from the use of pair programming. However, this has not yet been empirically evaluated. So, the current paper tries to determine which timing of introducing Pair Programming early in the first half of the semester vs. the second half of the semester is the most beneficial to the students learning outcomes.

3. EXPERIMENT DESIGN

The main goal of this study was to understand whether introducing pair programming in the beginning of the semester is more beneficial to students compared to introducing it at a later stage in the semester. The high-level question addressed by this paper was:

When would be the best possible time frame to get students involved in pair programming, in order to maximize the learning capacity of the students in introductory computer programming course?

In order to evaluate the learning efficiency we propose the following hypotheses.

Hypothesis 1: Introducing pair programming in the beginning of the semester significantly improve students learning ability as compared to using Pair Programming later in the semester.

Alternate Hypothesis 1a: Introducing pair programming later in the semester significantly improves students learning ability as compared to using Pair Programming at the beginning of the semester.

Null Hypothesis: Introducing pair programming at any stage are equally effective to improve the students' learning.

In order to test these hypotheses, a controlled experiment was conducted at NDSU computer science programming course, wherein ones section of the course used the Pair Programming at the beginning of the semester and the other section was introduced to the Pair Programming in the second half of the semester. Details of that study are presented in this section.

These experiments involved two types of learning methodologies: pair programming and a traditional method. It was decided to conduct the experiment in regular laboratory classes.

Here, the traditional method is referring to solo programming where only one student is involved in the development of a program for a laboratory exercise. Thus, the focus of the present study was to compare the learning efficiency of the students when they use pair programming in comparison to the traditional method for laboratory exercises.

Group A: The subjects belonging to this group worked in pairs and used pair programming to complete a programming assignment starting early in the semester. During the mid-way of the semester, the subjects worked individually on the remaining course work.

Group B: The subjects belonging to this group worked individually initially and were paired with a partner to complete the assignment for the reminder of the course.

The high level experiment design is further illustrated in the Figure 1 below.

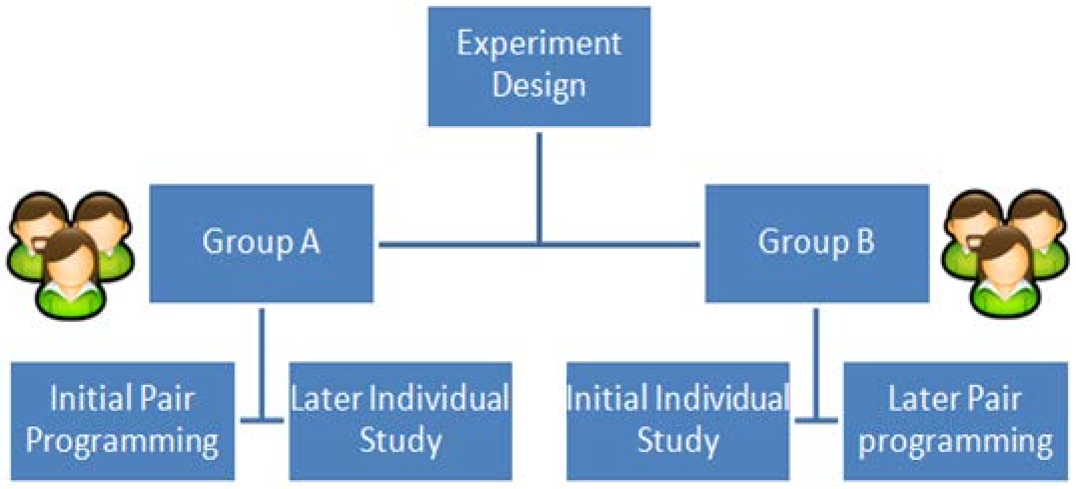


Figure 1: Experiment Design.

3.1. VARIABLES

In table 1, the independent and dependent variables are listed and discussed in the text below.

Table 1: Independent and Dependent Variables.

Independent variables	**Dependent variables**
Pair Programming method	Letter grade
Traditional learning method	Total grade
	Exam grades

Independent Variable: The independent variables represent the cause for the effect in any experiment. The two types of independent variables for our study were pair programming and traditional method. The outcome of our study is learning efficiency and the conjectured cause is the learning methodology. Thus, the independent variable for our study is learning methodology. In the present study, we were interested in manipulating the learning methodology to find its effect on learning efficiency and when should pair programming be introduced.

Dependent Variables: Researchers measured the following dependent variables:

Letter Grade: The final letter grade (i.e. `A', `B', etc.) received by the subject was used to evaluate their learning.

Total Grade: The final grade received by each subject included their grades on exams and programming exercises.

Exam Grades: The grades received on midterm and final exams.

3.2. STUDY SUBJECTS AND COURSE ASSIGNMENTS

The participating subjects in these study were 72 students enrolled in the CS1 Introductory Computer Programming at NDSU. Subjects in one of the sections of the course completed programming exercises in pairs, whereas subjects in the other section of the course completed programming exercises individually. The subjects in this study worked in pairs and individually throughout the semester and completed programming exercises related to variable assignment, evaluating conditions, string manipulation, looping, and other object oriented programming concepts during the semester.

Group A: The subjects were thirty-five undergraduate students enrolled in the CS1 course at NDSU.

Group B: The subjects were thirty-seven undergraduate students enrolled in the CS1 course at NDSU.

3.3. Study Procedure

The overall study procedure is illustrated in figure 2 below.

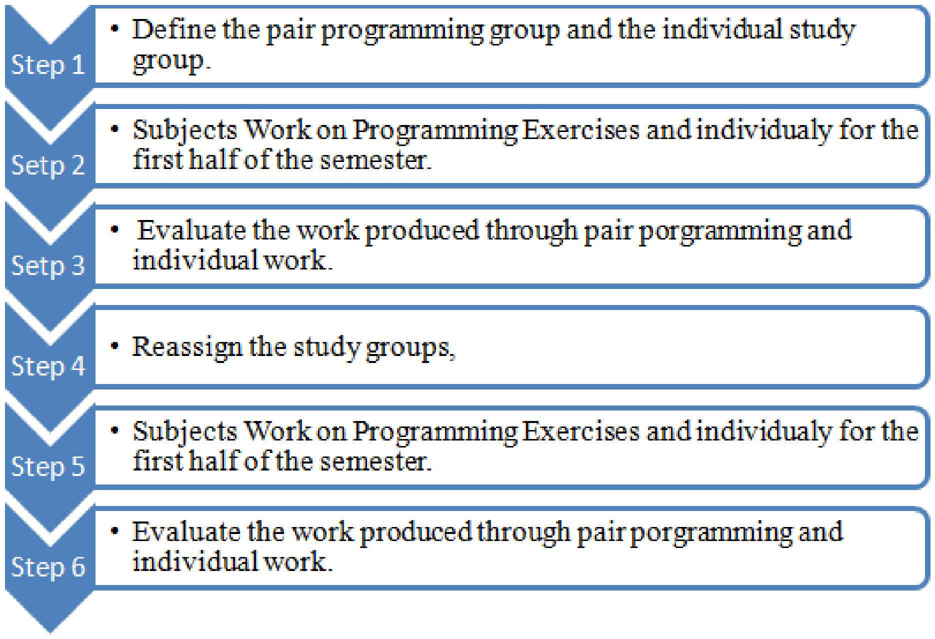

Step 1
Define the pair programming group and the individual study group.
Setp 2
Subjects Work on Programming Exercises and individualy for the first half of the semester.
Setp 3
Evaluate the work produced through pair porgramming and individual work.
Step 4
Reassign the study groups,
Step 5
Subjects Work on Programming Exercises and individualy for the first half of the semester.
Step 6
Evaluate the work produced through pair porgramming and individual work.

Figure 2: Experiment Procedure

Step 1: *Subjects were assigned to two groups*. In the experimental group (Group A), all subjects were paired with a partner to work on programming exercises early in the semester. The control group (group B) worked individually on programming assignments early in the semester.

Step 2: *Subjects Work on Programming Exercises*: In the Group A, all subjects worked on programming exercises with their assigned pairs for the duration of the first half of the semester, and then completed programming exercises individually during the second half the semester. In group B, all subjects worked on programming exercises individually for the duration of the first half of the semester, whereas the remaining section the subjects were paired and worked collaboratively on given programming assignments. This was done both to provide a control group, and to ensure that all subjects had the opportunity to use pair programming in the class.

Step 3: *Evaluate the subjects on group A (who used pair programming) and group B (who worked individually)* on their given programming assignments for the first half of the semester, based on the understanding basic programming concepts which was taught during the course.

Step 4: The study subjects will be redistributed so that subjects who started with pair programming will now be doing individual study and subjects who were studying individually will be now using pair programming.

Step 5:*Subjects Work on Programming Exercises*: In the Group A, all subjects worked on programming exercises individually for the rest of the semester, In group B, all subjects worked on programming exercises were now paired with another subject allowing them to work as pairs for the rest of the semester.

Step 6: *Evaluate the subjects on group A (who used pair programming) and group B (who worked individually)* on their given programming assignments for the first half of the semester, based on the understanding basic programming concepts which was taught during the course.

On the day prior to beginning the programming exercise, subjects were shown a ten minute video in class that described how to use pair programming effectively on their programming assignment [11]. The next day during their lab session, subjects in both groups worked on a programming exercise that was specifically designed by the instructor such that it should not take more than one lab period (fifty minutes) to complete. Subjects were given a programming exercise related to the topics that were being covered in class that week and were given two lab periods (with each lasting fifty minutes) and three days of out-of class time to complete the programming exercise. Subjects were only monitored while working on the exercise during the designated lab periods. Subjects worked with their assigned partner to complete the programming exercise.

Table 2: Student Allocation

	First half of the semester		**Second half of the semester**	
	Group A	**Group B**	**Group A**	**Group B**
Pair programming	35	0	37	0
Individual	0	37	0	35

In the table 2, the allocation of students for each half of the semester is described. Thirty five students were paired initially in the semester to work in pair for the first half of the semester in which they worked of programming assignments with pairs. During the second half of the semester those 35 students worked individually on programming assignments. Thirty seven students worked individually during the first half of the semester and then paired with a partner to work on the second half of the semester.

3.4. DATA COLLECTION AND EVALUATION CRITERIA

Student performances on lab assignments and tests (including mid-term and final examinations) were analyzed to study the effects of the mental-model-based pairings on the students' performance. The final course grades for subjects in the study were also analyzed so that we can determine the shift in student performance with related to the introduction of pair programming.

4. RESULTS

We analyzed the midterm and end-of the term examination marks for the subjects who used pair programming initially in the semester vs. subjects who started off studying individually. Some background on students' performance on these tasks, students generally did very well on the all examinations. Average grades for students in Group A who were introduced to pair programming at the beginning was 51.11 and 43.28 for midterm and final exams, respectively. Average scores for students in Group B who worked individually initially and then introduced to the pair programming later was 54.94 and 46.81 for midterm and final exams, respectively. Figure 3, the average test scores are compared for each group.

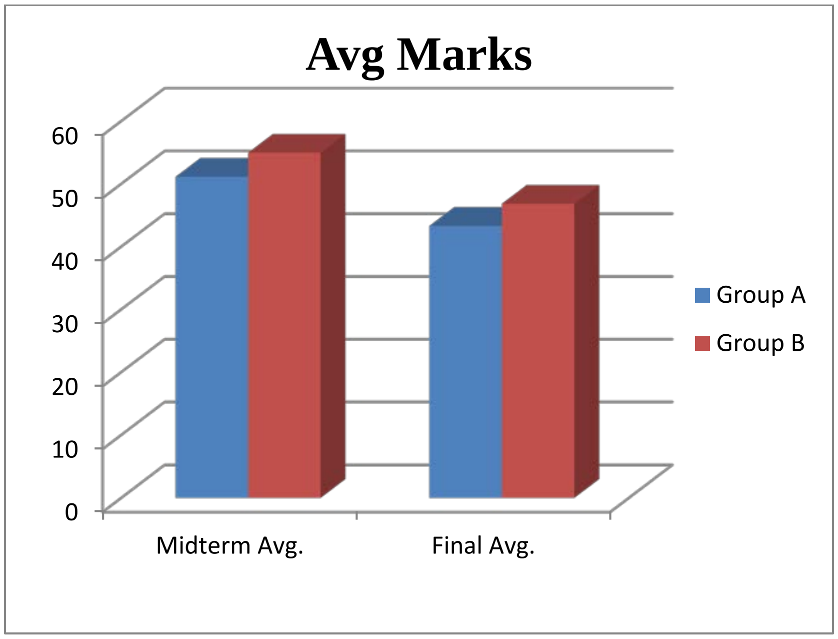


Figure 3: Average Marks

There is a non-significant increase in scores of students who used Pair Programming later in the semester in comparison to the students who used Pair Programming at the beginning of the semester. When you consider the midterm marks of Group A and Group B, the Group B subjects have performed better, and there is an increase of 3.83 for midterm which was not statistically significant. Similarly, there is a small increase of 3.52 for final scores of group B subjects when compared to group A subjects. Table 4 summarizes these results.

Table 3: Marks Comparison of Group A and B

	Average Scores		**Change in Midterm Versus Final Marks**
	Midterm Exam	**Final Exam**	
Group A	51.11	43.28	7.82
Group B	54.94	46.81	8.13
Scores difference between Group A & B	3.83	3.52	

When comparing the final scores as listed in Table 3, even though the Group B subjects had performed better than the Group A subjects, both groups did poorly in final exams. We have compared each group's midterm marks vs. the final marks. When considering Group A total of 7.8% drop in final grades compared to midterm grades. This is calculated by the difference between midterm marks and final marks in group A.

Difference between final/midterm marks is calculated as below.

$$Difference\ in\ marks = Midterm\ Marks - Final\ Marks$$

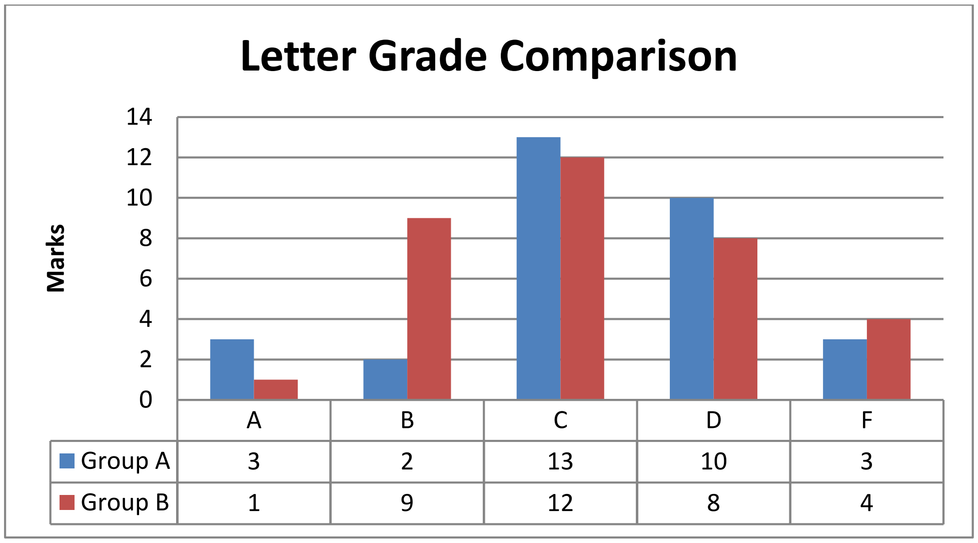
Equation 1 : Difference in Midterm and Final Marks

Group A ➔ 51.11-43.28 = 7.82

Group B ➔ 54.94 – 46.81 = 8.13

In group B this fall is about 8.1% drop in marks. The calculation of the drop in marks is as follows using the above formula 1.

This observation in final marks also validates previous research, which also concluded that pair programming does little to improve student learning and usually only in limited ways such as better grades on programming exercises [13] and generally does not extend to exams.



	A	B	C	D	F
🟦 Group A	3	2	13	10	3
🟥 Group B	1	9	12	8	4

Figure 4: Letter Grade Comparison

Figure 4 shows a graphical representation of grades achieved by each group. Based on these results, Group A had a larger number of "A" a total of 3 students out of 37 students obtained a grade of A, whereas in Group B students' performance is much higher than the Group A students performance.

Table 4: Final Grade Comparison

Letter Grade	Group A	% Group A	Group B	% Group B
A	3	9.677419355	1	2.941176
B	2	6.451612903	9	26.47059
C	13	41.93548387	12	35.29412
D	10	32.25806452	8	23.52941
F	3	9.677419355	4	11.76471

The comparison of letter grades between the two groups are as follows, in Group A 9.6% students obtained a grade A as compared to the 2.9% of students in Group B that obtained a grade A. These results show that the total number of students who got a grade C or higher was 18 in Group A in comparison to 22 students in Group B. In Table 4 the grades that each group obtained are summarized

Table 5: Mean & Standard Deviation of the Exam Marks

	Group A		**Group B**	
	Midterm	**Final**	**Midterm**	**Final**
STDev	12.29	20.202	14.22	17.74
Mean	51.11	43.28	54.94	46.81

Table 6 also shows the p-values of midterm and final exams.

Table 6: Introducing Pair Programming Initially versus Later

	Midterm	**Final**
P-Value	0.252	0.456

Results of the first pair and individual assessments initially indicated that there was a very small difference between pair and individual marks. A similar study by Radermacher [27, 30] also validates our finding in this study. The results from Radermacher et al.[2012] shows that students perceive pair programming as being beneficial and all of the subjects who used pair programming indicated that they would prefer using it again in the future as opposed to working individually. In addition pair programming is beneficial to student performance among students who may be struggling with the course. Research also indicates that pair programming allows a student pair to complete a given amount of work more quickly than individuals. [27]

5. THREATS TO VALIDITY

A major threat to validity deals with difficulties in being able to gather precise measurements of each student’s performance based on the time when pair programming was introduced. First, attendance was not taken during this course, so it is possible that whether or not subjects attended lectures had some influence in the overall scores of midterm and final by the subjects. The subjects of the experiments were mostly new to pair programming, and thus might perform worse because they were not adapted to the duration of short experiments, conduct so they may not have enough time to learn how to conduct pair programming well, as a result they could not improve statistically.

6. DISCUSSION AND CONCLUSION

This presents an analysis of the data to provide answers to the hypothesis posed in Section 3.0.The data analyzed includes the subject's the grade received on the marks obtained during midterm and final exam marks. Subjects' pair performances on midterm and final exams and their individual performances on tests (including mid-term and final examinations) were analyzed to determine if the introducing pair programming at the beginning of the semester is more beneficial to the student than introducing it at a later stage.

The results from this study do not indicate that one group is significantly more effective than any other. There was not much difference between the exams scores in group A and Group B students. However results showed that students perceive pair programming as being beneficial and all of the subjects who used pair programming indicated that they would prefer using it again in the future as opposed to working individually Pair programming introduced at any time of the course was beneficial to students. It also shows that pair programming is beneficial to student performance among students who may be struggling with the course.

The results of this study provides support to encourage further studies, particularly to increase the number of data points in order to determine when is the best time to introduce pair programming. Also it is possible for students to perform well in the initial weeks of the course and then struggle with more difficult topics. When pair programming was introduced in later stages, students had a better knowledge on basic concepts in programming and were able to demonstrate those skills better. This would have resulted in better overall performance. However, the results do not show a significant improvement when using Pair Programming later vs. earlier.

Although this study has provided some valuable insights, results should not be unconditionally generalized due to the small number of students who participated in this study. It is recommended that the study should be replicated and involve a larger sample of participant's studies over a longer time period. More biographical information could also be useful for analysis of differences between male and female, different personality types, and age groups.

7. REFERENCES

[1] L. Williams and R. Kessler, Pair Programming Illuminated: Addison-Wesley, 2003.

[2] L. Williams, R.R. Kessler, W. Cunningham, and R. Jeffries, Strengthening the case for pair programming, IEEE Software, 17(4), July – Aug. 2000, pages 19 – 25, 2000

[3] K. Beck, Extreme Programming Explained: Embrace Change: Addison-Wesley, 1999.

[4] P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, “New Directions on Agile Methods: A Comparative Analysis,” International Conference on Software Engineering, 2003.

[5] B. F. Hanks, “Tool Support for Distributed Pair Programming,” Workshop on Distributed Pair Programming. Extreme Programming and Agile Methods - XP/Agile Universe, 2002.

[6] Domino, M. A., R. Webb Collins &Hevner, A.R. (2007). Controlled experimentation on adaptations of pair programming. Springer.

[7] J. Chong, T. Hurlbutt, The social dynamics of pair programming, ICSE 2007 29th International Conference on Software Engineering, IEEE Computer Society, 2007.

[8] C. McDowell, L. Werner, H.F. Bullock, J. Fernald, The effects of Pair-Programming on Performance in an Introductory Programming Course, Proceedings 33rd SIGCSE Technical Symposium on Computer Science Education, ACM Press, 2002.

[9] C. McDowell, L. Werner, H.F. Bullock, J. Fernald, The impact of pair programming on student performance perception and persistence, Proceedings 25th International Conference on Software Engineering, 3 – 10 May 2003, pages 602 – 607, 2003.

[10] C. McDowell, L. Werner, H.F. Bullock, J. Fernald, Pair programming improves student retention, confidence, and program quality, Communications of the ACM, Vol. 49, No. 8, pages 90-95, 2006.

[11] D. Wells. The rules of extreme programming, 1999.

[12] Klemm,W.R. (1994). Using a formal collaborative learning paradigm for veterinary medical education. Journal of Vertinary Medical Education, 21(1).

[13] C. McDowell, L. Werner, H.F. Bullock, J. Fernald, The effects of Pair-Programming on Performance in an Introductory Programming Course, Proceedings 33rd SIGCSE Technical Symposium on Computer Science education, ACM Press, 2002.

[14] C. McDowell, L. Werner, H.F. Bullock, J. Fernald, The impact of pair programming on student performance perception and persistence, Proceedings 25th International Conference on Software Engineering, 3 – 10 May 2003, pages 602 – 607, 2003.

[15] C. McDowell, L. Werner, H.F. Bullock, J. Fernald, Pair programming improves student retention, confidence, and program quality, Communications of the ACM, Vol. 49, No. 8, pages 90-95, 2006

[16] E. Mendes, L. B. Al Fakhri, A. Luxton-Reilly, Investigating Pair Programming in a 2nd year Software Development and Design Computer Science Course, Proceedings of ITiCSE'05, June 2005, pages 296-300, 2005.

[17] E. Mendes, L. B. Al Fakhri, A. Luxton-Reilly, A Replicated Experiment of Pair Programming in a 2nd year Software Development and Design Computer Science Course, Proceedings of ITiCSE'06, 2006.

[18] N. Nagappan, L. Williams. M. Ferzli, E. Wiebe, K. Yang, C. Miller, S. Balik, Improving the CS1 Experience with Pair Programming, Proceedings of the 34th SIGCSE Technical

Symposium on Computer Science Education, ACM SIGCSE Bulletin, 35 (1), pages 359 – 362, 2003.

[19] S. B. Berenson, K. M. Slaten, L. Williams, C. Ho, Voices of women in a software engineering course: Reflections on collaboration, ACM Journal of Educational Resources in Computing, 4(1), 2004, pages 1-18

[20] S. B. Berenson, K. M. Slaten, L. Williams, C. Ho, Voices of women in a software engineering course: Reflections on collaboration, ACM Journal of Educational Resources in Computing, 4(1), 2004, pages 1-18

[21] C. Ho.; Slaten, K.; Williams, L.; and Berenson, S., Examining the Impact of Pair Programming on Female Students, NCSU Technical Report, TR-2004-20, June 17, 2004.

[22] D.C. Cliburn, Experiences with pair programming at a small college, Journal of Computing Sciences in Colleges, October 2003, 19(1), 2003.

[23] L. Williams, K. Yang, E. Wiebe, M. Ferzli, C. Miller, Pair Programming in an Introductory Computer Science Course: Initial Results and Recommendations,Proceedings of OOPSLA, 4th Nov. – 8th Nov. 2002.

[24] E.F. Gehringer, A Pair-Programming experiment in a non-programming courses, OOPSLA'03, ACM Press, 2003.

[25] S. Heiberg, Puus U., P. Salumaa, A. Seeba, Pairprogramming effect on developers productivity, XP 2003 Extreme Programming and Agile Processes in Software Engineering Conference, Proceedings (Lecture Notes in Computer Science), Springer-Verlag,Vol.2675, pages 215-24, 2003.

[26] Radermacher, A., Walia, G., Abufardeh, S., and Myronovych, O, Guidelines for Implementing Pair Programming in Introductory CS Courses: Experience Report” Proceedings of 2014 International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS). July 21- 24, 2014 USA

[27] Radermacher, A., Walia, G. "Improving Student Learning Outcomes with Pair Programming" Proceedings of the 8th International Conference on Computing Educational Research - ICER'2012. September 10-12, 2012 Auckland, New Zealand. pp. 87-92

[28] Radermacher, A., Walia, G. and Rummelt, R., “Assigning Student Programming Pairs Based on Their Mental Model,” Proceedings of the 43rd ACM Technical Symposium on Computer Science Education. February 29 – March 3, SIGCSE 2012: Teaching, Learning, and Collaborating. Raleigh, North Carolina, USA

[29] Radermacher, A., Walia, G. "Investigating Student-Instructor Interactions When Using Pair Programming: An Empirical Study," Proceedings of the 24th IEEE-CS Conference on Software Engineering Education and Training. May 22-24, CSEE&T 2011 Waikiki, Honolulu, Hawaii. pp. 41-50

[30] Radermacher, A., Walia, G. "Investigating the Effective Implementation of Pair Programming: An Empirical Investigation," Proceedings of the 42nd ACM Technical Symposium on Computer Science Education. March 9-12, SIGCSE 2011 Dallas, Texas, USA. pp. 655-660

[31] Radermacher, A., and Walia, G. S. 2011. Investigating the effective implementation of pair programming: an empirical investigation. In Proceedings of the 42nd ACM.

Technical symposium on Computer science education (SIGCSE '11). ACM, New York, NY, USA, 655-660.

[32] Radermacher, A., and Walia, G. S. 2011. Investigating student-instructor interactions when using pair programming: an empirical study. To be published in 24th Conference on Software Engineering Education & Training (CSEET ’11). IEEE Computer Society, Washington, DC, USA.

[33] Kim Man Lui ; Hong Kong Polytech. Univ., Hong Kong ; Chan, K.C.C. ; Nosek, J.T., The Effect of Pairs in Program Design Tasks , Software Engineering, IEEE Transactions on (Volume:34 , Issue: 2) , March-April 2008