Birzeit University

Faculty of Information Technology

Master of Computing

دراسة علاقة أداء الطلبة والبرمجة الثنائية كأداة للتدريس في مواد البرمجة: جامعة بيرزيت كدراسة

Investigating the Relation between Student Performance and Pair-programming Teaching Technique in Programming Courses: Birzeit University as a Case Study

Submitted by:

Dima Abd El-Rahman Taji

Supervised by:

Dr. Mamoun Nawahdah

2016 - 1437

Investigating the Relation between Student Performance and Pair-programming Teaching Technique in Programming Courses: Birzeit University as a Case Study

Submitted by:

Dima Abd El-Rahman Taji

Supervised by:

Dr. Mamoun Nawahdah

A thesis submitted in partial fulfillment of requirement for the degree of master of Computing - Faculty of Information Technology - Graduate Studies

Birzeit University

2016 – 1437

Birzeit University

Faculty of Information Technology

Master of Computing

Thesis Approval

Investigating the Relation between Student Performance and Pair-programming Teaching Technique in Programming Courses: Birzeit University as a Case Study

Submitted by: Dima Abd El-Rahman Taji

Student Number: 1115390

Approved by the thesis committee:

Dr. Mamoun Nawahdah _____

Dr. Yousef Hassouneh _____

Dr. Yousef Abuzir _____

Acknowledgement

I would like to begin by expressing my sincere gratitude to my supervisor, Dr. Mamoun Nawahdah for his help, support, and guidance throughout the preparation of this thesis.

My deepest appreciation goes to the students who participated in the experiments, for their tolerance, and cooperation.

Special thanks to Dr. Majdi Mafarja, with whom I had long discussions in the initial phases of preparing my thesis.

I would also like to thank my supervisors in Progineer and in Thomson Reuters, for their support, tolerance, and understanding.

To my friends, for lending me an ear when I needed to rant, giving me sound advice, and never giving up on me, I am grateful for you. Foremost, to my dear friend Rana Rishmawi, who is running around on my behalf to submit my thesis and paperwork.

Last, but not least, my deepest gratitude and appreciation goes to my family, for their support and encouragement, through the ups and downs of my journey to get here.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BZU	Birzeit University
Comp231	Advanced Programming Course
CS	Computer Science
CSE	Computer System Engineering
df	Degree of Freedom
GUI	Graphical User Interface
PP	Pair-programming
PPU	Palestine Polytechnic University
PTUK	Palestine Technical University – Kadoorie
RAD	Rapid Application Development
TA	Teaching Assistant
XP	Extreme Programming

Abstract

Pair-programming is a software development technique that was introduced as part of Extreme Programming. In pair-programming, two developer share a computer to work together on developing one piece of code. This technique started in the software industry, but was adapted and applied in some university courses where programming is taught to students. This method is highly controversial both in industry and in education, and has numerous advocates and as many critics. Believing in the merits of pair-programming, and to test its effects in a Middle Eastern community, we devised an experiment that was carried out over two semesters in Birzeit University. The experiment targeted two sections per semester of the Advanced Programming course. The students of one of the sections worked in pairs during the lab sessions, applying pair-programming rules and techniques. The second section had students who worked individually, as it is the norm in most programming labs. Video recordings were recorded throughout the lab sessions, and then studied and analyzed. In addition, code samples were collected from the students to study the effect of pair-programming on the students' code quality. Through this experiment we found out that pair-programming has the potential to increase the students' confidence, and their enjoyment of the course, and improved the course's completion rate. In addition, the students in the pair-programming sections showed that they were able to individually produce code of better quality than the students in the traditional section.

الملخص

البرمجة الثنائية هي احد تقنيات تطوير برمجيات الحاسوب. كانت بداية البرمجة الثنائية كأحد ممارسات البرمجة القصوى. تتميز البرمجة الثنائية بجلوس مبرمجين اثنين الى جهاز حاسوب واحد، والعمل سوية لكتابة البرامج. بدأ تطبيق هذه التقنية في أسواق العمل، ولاحقا تم استخدامها كأسلوب تدريس في مساقات البرمجة في الجامعات. لاقت البرمجة الثنائية النجاحات والإخفاقات بذات الوقت، لذلك لهذه التقنية العديد من المؤيدين والمعارضين، وما زالت احد المواضيع المثير للجدل في مجال هندسة البرمجيات. ايمانا منا بقدرة هذه التقنية على تحسين طرق تدريس البرمجة، ورغبة بدراسة تأثير تطبيق البرمجة الثنائية في مجتمع شرقي، قمنا بتطوير تجربة لدراسة هذه العوامل. تعرض هذه الاطروحة التجربة التي قمنا بها على مدار فصليين دراسيين، قمنا في كل منهما بتدريس مساق البرمجة المتقدمة لشعبتيين، احداهما عمل فيها الطلاب على شكل ازواج وقاموا بتطبيق مبادئ البرمجة الثنائية، والثانية عمل فيها الطلاب بشكل منفرد حسب الطرق المتعارف عليها بتدريس البرمجة. قمنا بتسجيل جزء من عمل الطلاب داخل مختبرات البرمجة، وتم دراسة وتحليل مقاطع الفيديو. بالإضافة الى ذلك تمت دراسة عينة من برامج الطلاب لدراسة نوعية البرامج التي قاموا بكتابتها. تم استنتاج ان البرمجة الثنائية لديها القدرة على زيادة ثقة الطلاب بنفسهم، مما حسن من نسبة الطلاب الذين قاموا بإنهاء متطلبات المساق، كما زادت من استمتاعهم بالمساق. بالإضافة الى ذلك، تم ملاحظة انه عندما عملوا بشكل فردي كان باستطاعة الطلاب الذين استخدموا تقنية البرمجة الثنائية كتابة برامج ذات نوعية أحسن من البرامج التي كتبها زملائهم الذين عملوا بمفرهم خلال الفصل.

Chapter 1 Introduction

Trends in teaching techniques in Computer Science courses are trying to bridge the gap between the university environment and work environment. In universities, students are usually pushed to do their work and assignments individually, and collaborations are most often considered as cheating attempts. When working in software companies and other institutes, team players are the ones that usually thrive, and produce better results. This pushed for incorporating collaborative work in the teaching of Computer Science courses in universities around the world, to facilitate the transition of Computer Science students from an individual-centered environment to a group-centered environment.

In 1999, the notion of extreme programming (XP) was introduced, and one of the more controversial practices it introduced was pair-programming. Pair-programming (PP) is *"the practice whereby two programmers work together at one computer, collaborating on the same algorithm, code, or test"*[1]. The emphasis of pair-programming being that both programmers are working on one device, to produce one program.

In PP, two programmers will sit next to each other on one computer, looking at the same screen, as shown in Figure 1. They use one keyboard and one mouse, to manipulate the computer and type their code. They work together to produce one program, or one piece of code. The programmers work together, taking turns to type, and continuously discuss their code, improve it, revise it, and debug it.

Figure 1 - Two software developores practicing PP[2]

The success that this technique found in the software development industry and due to its applicability in labs, this technique was attempted, as a teaching technique, in a number of universities worldwide [3-11]. The experiments where PP was applied in teaching produced various degrees of success, and concentrated on a wide number of parameters, and environment variables.

In this research we studied the effect that PP as a teaching technique has on the performance of students in a Middle Eastern class setting. The study found that PP had the ability to improve the course completion rate, the students' enjoyment of the course, and quality of the code that they produced by the end of the semester.

1.1 Motivation

Introduction to programming courses are reputed with having low averages, with failure rates varying between 30% to 50% worldwide[12]. In another study that was carried out in the University of West Indies, the average fail rate was 20% in the period from 2004 and 2009[13]. In addition, students who fail to grasp the

fundamental concepts of programming in the first introductory courses are often unable to recover and catch up, and end up dropping out of Computer Science programs[13, 14]. This has been noted as well in Computer Science (CS) and Computer System Engineering (CSE) students at Birzeit University (BZU), where the fail rates during the two semesters previous to our experiment were 29% in the first semester of 2013 – 2014, and 42% in the second semester of the same year.

Literature shows that the application of PP as a teaching technique has promising results of improving the education of computer science, if applied correctly[4]. Studies presenting experiments where PP is applied as a teaching technique show an improvement in the students' programming ability, and learning experience. In an attempt to acclimate students to working within a team, and to improve their learning abilities, and their programming quality, we applied PP as a teaching technique to the Advanced Programming course (Comp231) offered by the CS Department in BZU.

Even though similar experiments in the application of PP in education were carried out, it has not, to our knowledge, been applied in a Middle Eastern society so far. A Middle Eastern society has restrictions on gender interaction, and social behaviours norms. This may be a limitation to the extent of interaction between students in the same class.

Being a conservative society, a Middle Eastern society generally, and the Palestinian society specifically, does not look favourably on pairs formed of students from different genders. Therefore, male students often prefer to work

with other male students, and female students with other female students. This usually limits the exposure of a student to his or her peers, which leads to limiting their experience and knowledge.

In addition, from our experience with teaching students in BZU, it was noted that students regard any group activity as an opportunity to have the workload fall on one or more students, but not the entire group. However, PP's success depends on having both members of the pair work equally to obtain the best results.

Having students understand that they will have to work with each other, regardless of who their partner is, or where they are from, was an obstacle that we had to overcome. This makes the experiment conducted in BZU stand out from other experiments, due to the social norms that distinguishes the Palestinian and Middle Eastern society from other societies.

1.2 Question

Investigate the effects of used teaching techniques over the performance and behaviour of computer programming students.

1.3 Objectives

- Determine if PP can allow the students to write quality software, in terms of line of codes, comments, and code complexity and correctness.
- Determine if PP can improve the performance of the students in Comp231 courses, in terms of grades and general averages.

- Determine if PP can increase the students' completion rate in Comp231 courses.
- Determine if PP can increase the enjoyment, measured by the amount of smiling and interaction during the labs, and the feedback from the students.

1.4 Hypothesis

We hypothesize that:

- PP can improve the quality of the codes produced by students, because the code is constantly being revised by another person, and the solution discussed and improved by the two people in the pair, as shown in [7, 15, 16].
- PP can improve the performance of the students in the introductory and advanced programming courses, in terms of grades, and software quality because students tend to learn better from their peers[17].
- PP can increase the students' completion rate because PP promotes self confidence as noted in [8, 14, 15, 18], and allows for the students to catch up on concepts they might have missed throughout lectures.
- PP can increase the enjoyment of students in introductory and advanced programming courses, as per [14, 15]. This is dues to the fact that PP allows for more social interactions during the labs, which usually are long and tedious.

The null hypotheses that will be tested in this research are:

- H_{0_1}: PP has no effect on the quality of the code that the students produce.
- H_{0_2}: PP has no effect on the grades that the students get.
- H_{0_3}: PP has no effect on the students' course completion rate.
- H_{0_4}: PP has no effect on the students' enjoyment of the lab sessions.

1.5 Methodology

The research methodology was divided into five phases:

- Studied the literature about the application of PP in education sector and the software development industry.
- Conducted an experiment over two sections of Comp231 course in the CS Department in BZU during the first semester 2014 - 2015, where one section is a PP section, and the second is a traditional section. This entailed the collection of data, through questionnaires and personal interviews with the students, video recordings and images taken during labs, and the observation of the students' interactions during their labs, as well as their grades throughout the course.
- Confirmed the results of the first experiment by repeating the PP experiment during the second semester 2014 – 2015, while collecting the same kind of data as that that was collected during the previous semester.

- Studied the results obtained from each of the semesters separately, and in comparison to each other.
- Produced recommendations and suggestions regarding the application of PP as a teaching technique.

The research relied on surveys and experiments that were carried out in BZU, as well as studying the published literature on this topic in various countries. In addition, the data collected during the labs, whether through video recordings or images was analyzed using appropriate software, and tests. The quality of the code produced by the students during the PP labs was studied and compared to code produced in other labs. The students' behaviour, and their interactions were observed and noted. From the results of these surveys and research, the applicability of PP as a teaching technique in Palestinian universities was determined, and recommendations were made accordingly.

1.6 Research Achievements

The findings of this research were presented in two conferences concerned with innovative teaching in higher education institutions, and received a generally positive feedback from the attendees of both conferences.

The first conference was "The First national conference in teaching and learning in higher education: Towards Creative Initiatives in Teaching and Learning in Higher Education". The conference was held by the Center for Excellence in Teaching and Learning in Palestine Polytechnic University (PPU), on the 2nd and

3rd of December, 2014, where we presented a paper titled "Enhancing Computer Learning Activities Using Pair-Programming Techniques" [19]. Figure 2 is a photo from the presentation.

Figure 2 - A picture from the presentation of our research in PPU

The second conference in which this experiment was presented was the "Innovation in Teaching and Learning: From Policy to Practice". It was organized by the Kadoorie Center for Learning and Teaching Innovation in the Palestine Technical University – Kadoorie (PTUK), between the 14th and the 16th of December 2014.

The response that the research received was that the idea was applicable, and had several potential benefits that included reducing the number of needed devices in the labs, as well as helping the students integrate better in the work environment after graduation.

Nevertheless, some of the comments were directed towards the methodology followed to select the samples and analyze the collected data. A number of

statistical analysis tests were suggested such as the T-test and F-test. These tests where studied, and the relevant ones were selected and applied to the data.

Two papers based on the research done in this thesis were presented in the 2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), in China. Another paper will be presented in the 2016 IEEE Global Engineering Education Conference (EDUCON) in the UAE.

1.7 Thesis outline

In chapter 2 of this thesis, we present a detailed background and literature review. We review how software development models progressed, and the introduction of extreme programming and the PP technique. Following that, we go into how PP is being applied both in the industry and in education, and its characteristics, merits, and drawbacks in both fields. In chapter 3, we detail the methodology followed in the writing of this thesis. This includes both the qualitative and quantitative data collection, and analysis, as well as the field experiment setup. We present the results and discussion of the experiment, and its findings in chapter 4. Finally, in chapter 5 we present the conclusion, and future work.

Chapter 2 Literature Review

This chapter aims to present the literature that has been done on the topic of PP. It starts by presenting a brief history of how the software development process models progressed, and the introduction of PP as part the agile model, and the XP movement. Following this, the chapter goes into presenting the application of PP in the software industry, and its merits and drawbacks. Finally, it presents PP in the education, its merits and drawbacks, and explains how experiments similar to the one presented by this thesis were set up and carried out in different universities.

Programming has always been considered a solitary activity[20]. This was the way it was taught, and the way it was practiced until 1999, when Kent Beck created extreme programming, and listed PP as one of its twelve practices[21]. Solo programming was associated with the waterfall development cycle in software engineering, suggested in 1970, in which a development team would meet with the customer once to get all the requirements of the system, and then design it, and implement its design[22]. Nowadays, most software development institutes give a lot of importance to the ability to work in teams, and research is always looking for methods that will improve the programmers' efficiency, productivity, and quality of work.

2.1 Progress of Software Development Process Models

Software development process models describe the stages that a software goes through during its lifecycle, including its design and production. Process models are continuously developing in order to achieve the highest level of efficiency possible. There are seven main software development process models[23]:

1. Waterfall model: this was the first software development model to be introduced. As illustrated in Figure 3, its basis is that each phase of development must be fully completed before moving on to the following phase.

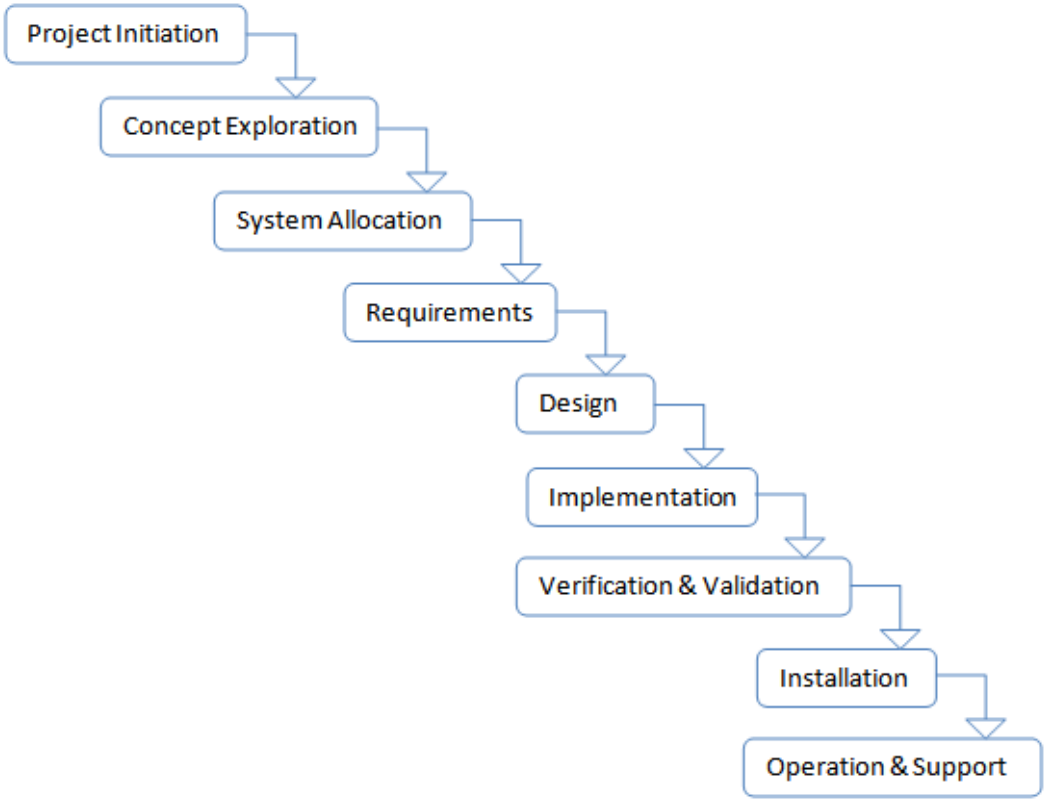


Figure 3 - Waterfall Model[22]

2. V model: V stands for Verification and Validation. This model is similar to the Waterfall model in that every phase must be completed before moving on to the phase that follows, but it differs, as shown in Figure 4, in that the testing of the product (the verification and validation) is done in parallel to the production phase.

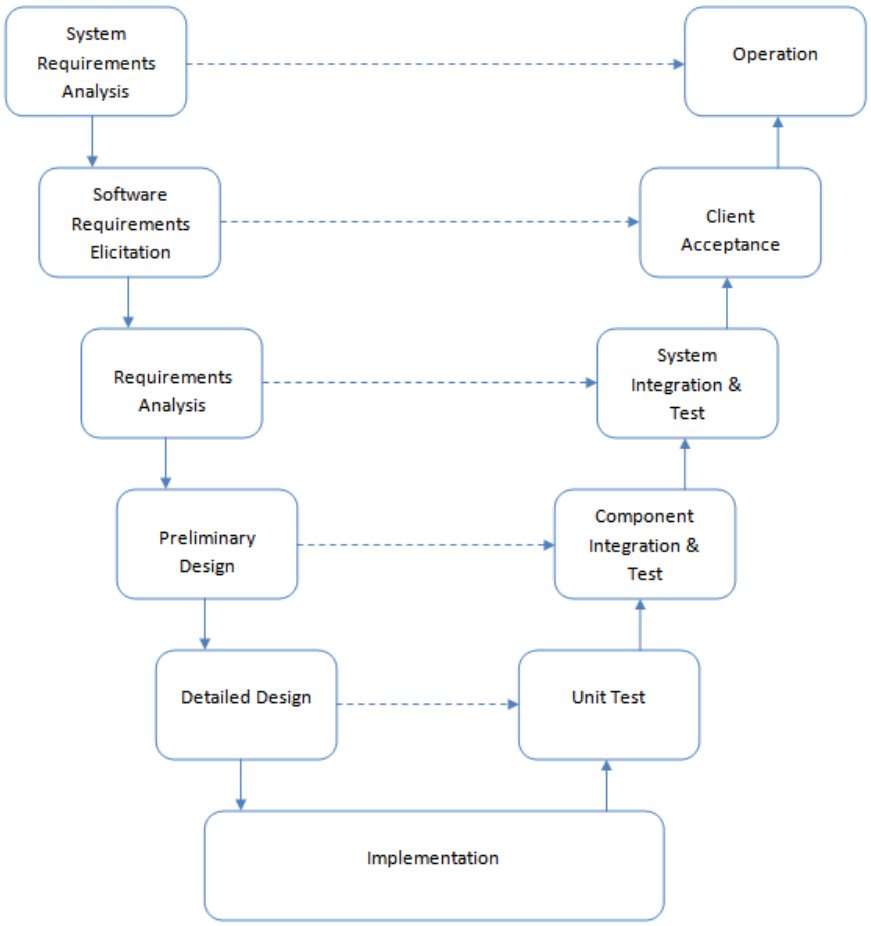


Figure 4 - V-Model[22]

3. Incremental model: This model divides the project into smaller modules, as in Figure 5. Each of these modules is developed using a Waterfall model.

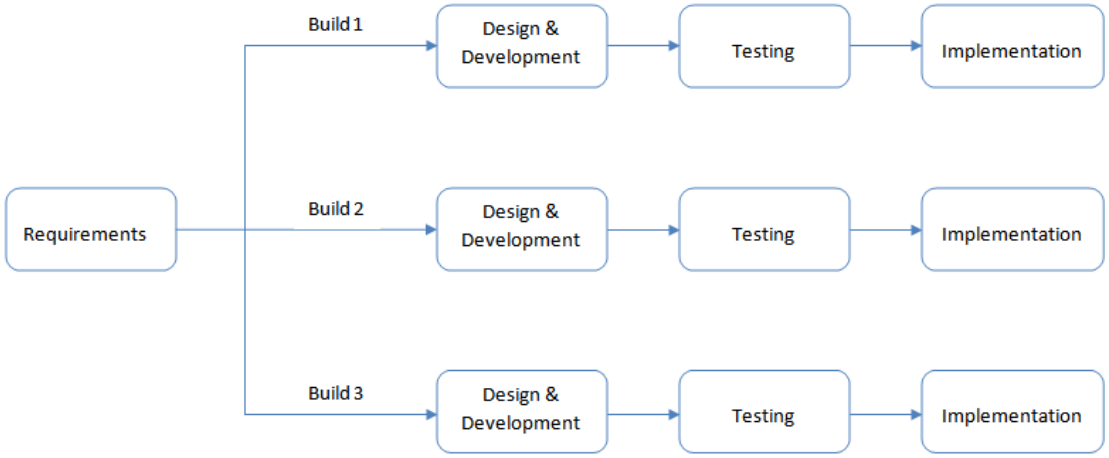


Figure 5 - Incremental Model[23]

4. Rapid Application Development (RAD) model: A type of the Incremental model, where every module can be produced separately and assembled at completion, as per Figure 6.

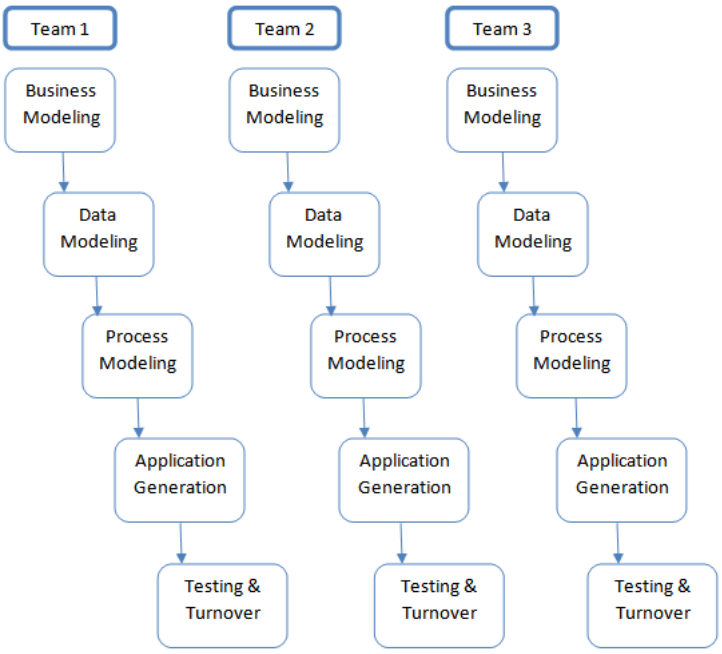


Figure 6 - RAD Model[23]

5. Agile model: Another type of the Incremental model, where the product is divided into releases, each release being developed and complete tested, with each functionality building up on the previous one. Figure 7 presents one release cycle.

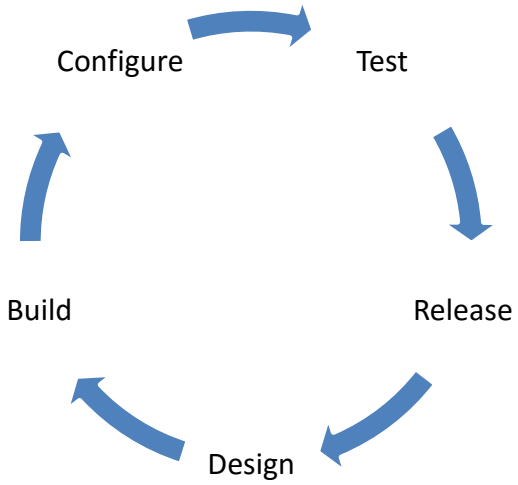


Figure 7 - Agile Model[24]

6. Iterative model: This model implements part of the software at each phase, and builds on that part in order to reach the complete specifications of the product. Figure 8 illustrates a three-phase project.

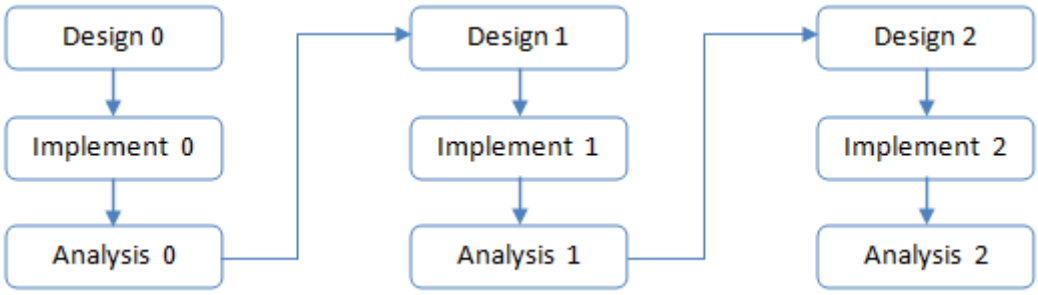


Figure 8 - Iterative Model[25]

7. Spiral model: A type of Iterative model, where each phase is divided into four parts; planning, risk analysis, engineering and

evaluation, as shown in Figure 9. At every spiral (iteration) of the project, the product goes through all four phases.

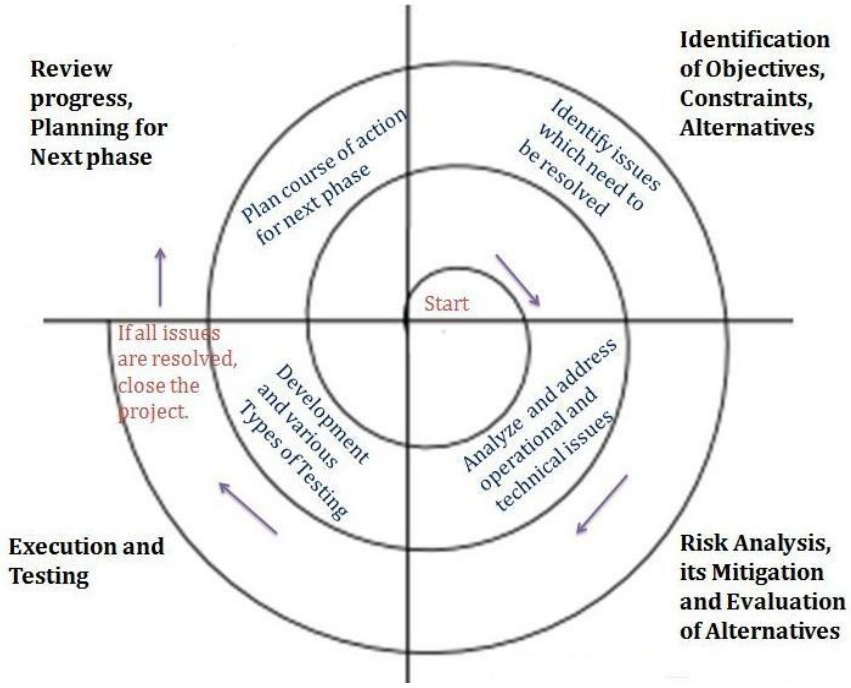


Figure 9 - Spiral Model[26]

It is noteworthy that the progress of software development models is continuously veering towards producing software than can be showed to the customer quickly, even if in phases. In addition, the emphasis is more on the customer and the people in relation to the software rather than process of development.

The waterfall model was proving to be less than effective, since the customer tends to change their mind continuously, and are often vague on what they want or need of their software[21]. This usually led to a significant increase in the cost of developing software, and prompted searching for alternative models that would

replace the waterfall model, and produce solutions to its shortcomings[27]. Moreover, solo programmers tend to make mistakes in their code that are not caught until the testing and debugging phase of a project[20], and their knowledge of the system as a whole is not concrete, with each developer understanding his own piece of the system only[20, 27]. This lead to the suggestion of the agile software development model.

2.1.1 Agile Software Development

Agility may best be defined as "*the ability to both create and respond to change in order to profit in a turbulent business environment*"[28], which is an aspiration to most software development teams. Judging from this definition, as well as the Manifesto for Agile Software Development[29], flexibility is a requirement to successful software development. Many practices are being adopted by software development teams in order to achieve the concepts of flexibility, interaction, and collaboration, among others. Of the practices listed by Kent for extreme programming, PP appears to stand out as a different, and somewhat controversial practice.

2.1.2 Pair-Programming

PP may be defined as "*the practice whereby two programmers work together at one computer, collaborating on the same algorithm, code, or test*"[1]. Not only do the programmers work at one computer, but "*all production code is written with two people looking at one machine, with one keyboard*"[30].

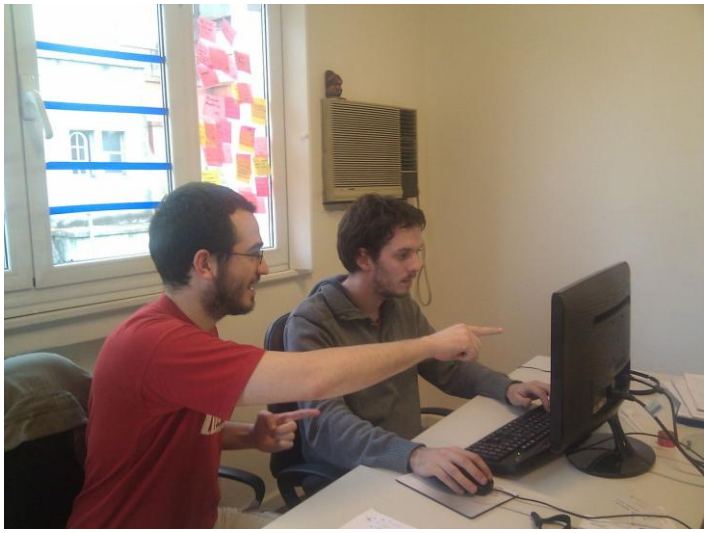


Figure 10 - A pair of programmers working with PP[31]

A programming pair consists of a driver and a navigator. In Figure 10, the person on the right is the driver, while the person on the left is the navigator. A driver's task is to actively type the code, and handle the keyboard, mouse, and any other input devices that are relevant. The navigator has to follow up with what is being typed on the screen, to catch any syntax mistakes, errors, or shortcomings of the code, in order to be able to correct and suggest better methods and solutions.

A key point that distinguishes the PP practice is that the pair should always switch roles. PP is most beneficial when this happens regularly. When the two members of the pair each assume a role and stick to it for an extended period of time, the efficiency of this technique decreases, and becomes less apparent.

Reviewing the literature, it is evident that many people can argue for PP, listing all the benefits that can come out of this practice. Nevertheless, a significant

portion of software developers and team leaders are against it, and can come up with probably as many hindrances[1, 20, 32, 33].

Some of the advantages of PP that are repeated throughout the literature include improving design quality[1, 3, 7, 20, 27, 34], reducing defects[20, 27, 34], contributing to pair members' skills[9, 20, 35], improving team communications[20, 27], and resulting in simpler code that is easier to extend[20]. In addition, some researchers have found that people working using PP tend to spend more effort on the tasks they undertake[5].

2.2 Pair-Programming in the Industry

A survey in 2007[36], that surveyed 781 software developers in the United States, stated that 69% of developers indicated that agile methods are being used in their institutions, in addition to 7.3% stating that these methods will be introduced in the following year. The reason to this popularity according to the survey is that on average 75% of agile projects are successful.

The applicability of PP as a practice according to the survey was rated by programmers at 3.4 on a scale of 5. Other research [37] indicates that PP's contribution to improving software correctness and development speed is at 10 – 15%.

Even though these researches have different designs and methodologies, which makes it difficult to indefinitely conclude that PP outperforms individual

programming. [33], this section attempts to present the views of researchers on the merits and the drawbacks of PP when applied in the industry.

To understand whether or not PP has a potential to succeed in the Palestinian industry, it was crucial to know first if the workers in the industry applied the PP technique in their projects or not. An email was sent to 103 institutions in the West Bank area. The email addresses where obtained from PITA's email directory. It was directed at the CEOs, directors, and general managers of these institutions. The email asked the recipients whether they PP in any of their projects. From these emails, we have received 36 replies, which constitute 34% of the sent emails. The full list can be found in Appendix I, and an email sample is added in Appendix II.

Table 1 – Using PP Techniques in Local Industry Projects

	Number	Percentage
Yes	8	22%
No	19	53%
Others	9	25%

Table 1 presents the statistics that were collected from this email. These statistics illustrate that only 20% of the institutes who replied to the sent email applied the PP technique to their projects. Around 10% of the institutes that answered "no" mentioned that they did not believe that PP was an efficient technique. The answers added under the Others entry consisted of replies from institutes that did

not have a software development department or group, or that did not want to answer the question.

2.2.1 Merits of Pair-Programming in the Industry

PP has many advantages that have been repeated throughout the literature. One of these advantages are improving design quality[1, 3, 7, 20, 27, 34, 38, 39], since two programmers collaborate together on the same piece of code, resulting in improving the quality of the design. Another advantage is reducing defects[20, 27, 34, 38, 40]. This is the result of continuous revision of code as it is being written, which makes it simpler to catch bugs and errors before the development phase is done.

Another advantage of using PP technique is improving team communication[20, 27, 38]. Since programming is usually viewed as a solo activity, there is not much communication going on between team members. Nevertheless, since PP involves having two programmers sitting at the same machine working on the same piece of code, they will have to communicate with each other. This has an effect of improving the social skills of team members and making their communication together more efficient.

An effect that has been noticed as well is that the code becomes simpler and easier to extend[20]. When two programmers collaborate on writing the same piece of code, obscure segments will be discussed and improved as a result of this collaboration. In addition, the resulting code will be more extendable than code

that only one person writes for the same reason of having more than one person writing it. In addition, some researchers have found that people working using PP tend to spend more effort on the tasks they undertake[5], which can be attributed to having another developer continuously reading the code and revising it.

2.2.2 Issues with Pair-Programming in the Industry

As with most techniques, PP has its adversaries. The first and most common issue that is brought up by most programmers who do not support the use of PP is the cost. Programmers and, more importantly, managers see PP as paying two people to do the work of one[1, 20, 41].

Individual differences might have a negative effect on the pair dynamic. Different personalities can clash and therefore be disruptive to the production process[1]. Therefore, managers have to be careful when pairing programmers together in order to avoid any problems in their teams[32].

Even if the differences were not in personalities, there might be differences in programming styles[1], and the desire of programmers to keep their code "private" [20] that might prevent the software development from running smoothly.

Personal differences emphasize the problem of distributing tasks. Distributing tasks can be an issue to the different tasks typologies[33]. This means that some tasks can be divided into smaller tasks that can be later on distributed to teams and

pairs to work at, while others are tasks that have to be dealt with as a whole. With different personalities and skills, the distribution of tasks becomes a more tedious task, where managers have to be careful to give each person a suitable task to their personality and skills[32].

Clashes between pair members can be enhanced by their inability to communicate together to solve even small issues[1]. This might be stemming from the fact that programming is taught as a solo activity in the first place[20], making communication skills a secondary skill that are not a priority to implement and enhance during the education phase.

For this reason, it is inevitable to introduce this technique in education, in order to measure its affectivity, and to prepare programming students to work in environments where PP is a key practice.

2.3 Pair-Programming in Education

Research shows that when students take a programming course, they are usually in their first year of university, when it is important to get them used to help them gain the skills needed during their university years as well as integrating into the industry[4]. This is why most experiments carried out with PP as a teaching technique is applied to introductory level courses[4, 6, 9, 17]. However, a number of researchers attempted to carry out the experiments on second-year and even advance students as well[7, 11].

This section begins by explaining how the pairs are formed in the computer science labs that use PP and how these labs are carried out, and what types of activities take place. Then, the way the assessment of the students' work is explained. Finally, it presents some of the merits and issues of PP as a teaching technique according to previous literature.

2.3.1 Experiment Setup

The way students are paired could affect the dynamics of the pair, and might result in an effect on the outcome of the experiments. The methods followed for the formation of pairs differed among experiments. They varied between forming pairs according to their level of programming experience[4, 6, 7], random formation[6, 11], letting students form their own pairs[4, 6], or a combination of having students select a number of potential partners, and then assigning one of them according to experience[9].

Teague explains that when students chose the partner with which, they often realized that this was not the partner with whom they actually wanted to work, or were most compatible[4]. This is why most researchers prefer having students paired according to their level of programming experience. This is usually decided according to the students' performance and grades in previous related programming courses, or assessing their programming experience[4, 6]. Nevertheless, it can be noted that other researchers claim that the experiments succeed most when students are paired randomly[11].

One of the most critical things when starting a PP experiment is identifying the difference between PP and team work, in the sense that PP does not mean each member of the team being responsible for completing a piece of the project individually and then combining pieces together[4].

In addition, pairs were instructed to constantly switch roles between driver and navigator[4, 6, 17]. The driver's task was to control the mouse and keyboard, as well as compiling and running the code. On the other hand, the navigator constantly looked for syntax and logical errors, and searched for resources and alternatives to the implementation. It was observed that in the later parts of the experiments this switch was taking place more frequently and smoothly than at the beginning[4].

2.3.2 Work Assessment

To evaluate their experiments, researchers took into consideration several aspects. In all the reviewed experiments, students were required to finish a number of assignments for which they were graded[4, 6, 7, 9, 11]. In addition, at least a final exam was given to assess students' performance in the course[9, 11].

Khan et al. [6] adopted an approach where they would award the submitted project giving both students the same grade, but incorporating a peer-review element, where each partner had to evaluate their partner's cooperation and performance on a scale of one to five, which factored in their final grade. On the

other hand, researchers such as Inoue in [42] opted for recording students while they were working in pairs, and analyzing the videos for certain parameters.

A number of researchers collected feedback from their students as well, through questionnaires and surveys[4, 6, 7]. These questionnaires aimed to measure different parameters, such as students' confidence level after PP, their perception of their compatibility with their partner, the effect of PP on their understanding the exercises and course material, and their enjoyment of the course.

2.3.3 Merits of Pair-Programming in Education

The merits of PP that have been observed in the industry pushed its implementation as a teaching technique. When used as a teaching technique, PP has several advantages in addition to those observed in the industry. Students' performance was shown to have improved when using PP[3, 10, 17], even when students were required to program individually for their final exam[9, 43]. Working in pairs promotes peer tutoring. Research shows that student may be more receptive to information when coming from their peers as opposed to an instructor[6].

In addition, students produced better quality programs than when working on their own[3, 8-10, 17]. Students exchange allows them to learn from each other's experiences[17], and brainstorm before working[4, 44], therefore allowing them to come up with the most efficient techniques to solve a given problem.

An advantages that has been noticed when surveying students who studied in labs applying PP technique is course enjoyment[3, 8-10, 17]. Students reported that they enjoy working in pairs more than they do on their own. This might be due to the social aspect of PP, or even due to the fact that they are producing good code and solving problems with a little less effort. On the same note, PP allows for less frustration when programming[4], since it is less likely to get stuck at a problem for long.

Studies have also show that applying PP in courses help improve course completion rate[3, 8-10, 17], where less students were found to drop the course in classes where PP is being practiced. This comes as a logical consequence to having a course where they are enjoying the work they are doing, while achieving good results.

2.3.4 Issues with Pair-Programming in Education

Applying PP as teaching technique shares some of the issues of applying PP in the industry. A challenge that faced instructors when assessing assignments submitted by a pair was how to distribute the weight among both students submitting the assignment. This was addressed in [6], where it was claimed that interviewing the students individually was time consuming, and thus not favoured.

It is expected of students to try to do what seems easier to them, or what helps them get the highest grade possible. It was noted by Khan et. al.[6] that based on their surveys, 24% of students chose to split the task between themselves, so that

each would work on a part individually, while 22% of students mentioned that they each did the task separately and submitted the better work.

Simon and Hank in [43] interviewed students who worked using PP, and one drawback of this technique that was expressed by these students is the high degree of dependence on one member of the pair to do the difficult and complicated tasks, which was also an issue presented by Gupta et. al. in [15]. From another perspective, some students felt that their partners were familiar with more advanced concepts that they found hard to keep up with.

As with pairing programmers in the industry, Gupta et. al. in [15] mention that a threat in PP is pairing the right students together and understanding their personalities, which imposes more work on the instructor than if they were to work individually.

In an experiment carried out by Wood et. al. [14] results show that if students didn't show an improvement in their work at the beginning of the application of PP, it was less likely that the technique will have a positive effect on their grades and their understanding of programming concepts. In addition, 11% of the students who underwent the experiment had their grades and programming level drop during the course of the experiment.

Chapter 3 Research Methodology

This research is focused on the study of the effects of pair-programming on the education of computer programming courses. It attempts to present, through a field experiment, the effects of applying pair-programming on the Comp231 course in BZU.

Even though many students in BZU turn out to be good programmers, it was noted by the instructors and teaching assistants (TAs) that a large number of them do not have a lot of confidence in their programming skills. In addition, students generally stick to what is being taught during the lecture, and very few of them attempt to try something different, such as a different approach, or a different algorithm, or a different data structure, from those explained during the lecture.

In addition to that, students often find lab sessions, which are a three-hour session each, to be long and tedious. They try their best to get out of the lab, and skip attending the session if they can. This ends up with a high withdrawal rate by the end of the semester, as well as a high fail rate.

3.1 Experiment Design

The experiment that is detailed in this thesis is an experiment between subjects. This experiment is described in Experimental Research[45] as the experiment where a participant can be part of either the control group or the treatment group

but not both. This means that in our experiment, the students will not be changing sections between the PP section and the traditional section.

According to MacKenzie[46] an experimental method requires having a manipulated variable and a response variable. The manipulated variable is defined as "a property of an interface or interaction technique that is presented to participants in different configurations."[46] The manipulated variable in this experiment is the PP technique used in teaching one of the sections.

The response variable is "a property of human behaviour that is observable, quantifiable, and measurable."[46] In this experiment, there are several response variables are being measures, that are the quality of the code the students produce, the scores that the students achieve in the course, the students' completion rate of the course, and their enjoyment of it.

The reason this experiment design was selected is that it allows the experiment to be carried out "with very little contamination by extraneous factors"[45]. This means that, aside from personal differences between students that are present in every section, it is safe to assume that any difference in results between the two groups of students will be a result of the different teaching technique that is being applied.

In an attempt to try to improve the level of programming students, in terms of their grades, their confidence, their pass rate, and their enjoyment, in BZU, we applied PP methods in the teaching of the students of the Comp231 course. This

course is preceded by an Introduction to Programming course that is taught in C, which gives the basic of procedural programming. The Comp231 course is offered for sophomore-level students, and covers the basics of Object-Oriented programming and is taught in Java. The course includes two one-hour lectures, and one three-hour lab per week, and spans over a 15-week semester.

The experiment was carried out twice during each of the fall and spring semesters of the educational year 2014 – 2015. In each semester, the experiment was conducted on two sections of the Comp231 course, one of which was a PP based section, and the other was a control section where the course was taught conventionally.

Figure 11 - Students Working Individually During a Lab Session in the Traditional Section

In both semester, both sections contained a mixture of students from CS, CSE, and other specializations. The sections were taught by the same instructor and TA. The lectures were given to both sections using the same methodology, and PP was

applied only during the lab sessions. Figure 11 shows a side of the traditional section, and Figure 12 shows a side of the PP section.

Both sections in each semester contained the same number of students. In the first semester, both sections had 30 students each, and in the second semester, both sections had 29 students each. Because all those parameters were identical, we were able to select the PP section randomly in both semesters with a coin toss.

Figure 12 - Students Working in Pairs During a Lab Session in the PP Section

We started the experiment after the first month of classes. The experiment was delayed until the fifth lab because it was imperative that the students get a feel of working individually, in order to grasp the essential concepts of object-oriented programming. After the first four labs were over, the students in the PP section were given a short presentation about PP and how it is applied.

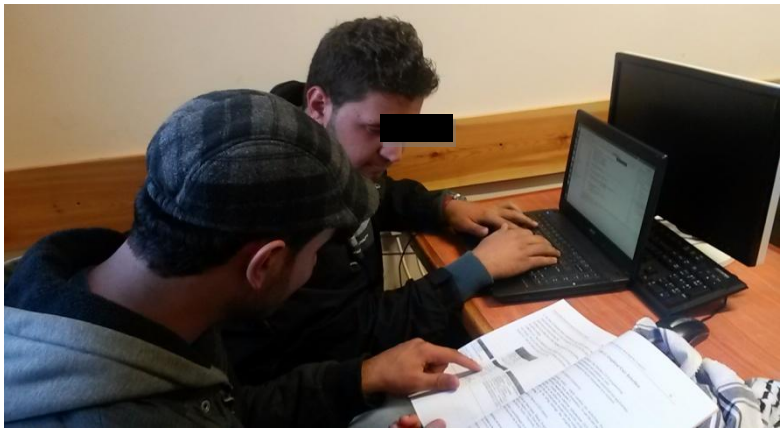


Figure 13 - Two Students Applying the PP Technique during a Lab Session

The students were asked to work in pairs during the lab only, as shown in Figure 13. They were presented with the programming problems in their lab workbook, and asked to solve them while working on one computer. Students were instructed to switch roles constantly, which usually happened in between exercises. Students were also encouraged to discuss the problem before starting to solve it, and to avoid asking the instructor of the TA for help, unless they both fail to reach a solution.

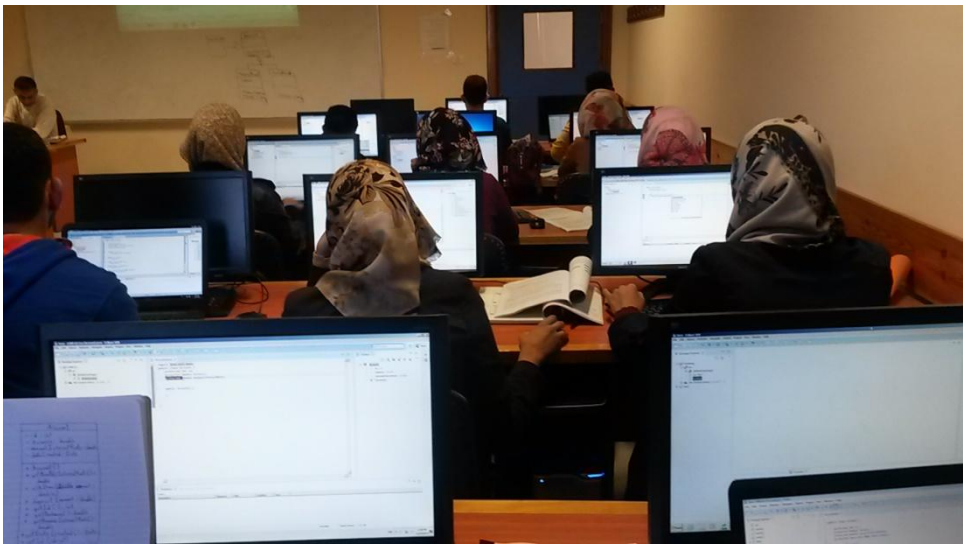


Figure 14 - Students Working Individually during a Lab Session in the Traditional Section

The control section, shown in Figure 14, where PP was not applied was not given any specific instructions. They continued with the lab sessions regularly after the fourth lab, with every student working on the assigned exercises on their own. Like the students of the PP section, they were encouraged to try to reach the solution on their own.

3.1.1 Pair Formation

When studying the literature, two main methods of pair formation stood out for having more merits than others; random selection, and students selecting their own partners. As the experiment was spanning over two semesters, both methods were tried, in an attempt to determine which was more beneficial to the participants.

During the first semester, students were asked to select their own partner for the duration of the semester, following the methods illustrated by Teague[4] and Khan[6] in their research. As most of the students knew each other for around a year, they opted to select a friend rather than a work partner. On the other hand, students who did not have friends in the same section ended up in random pairs.

During the second semester, the students were distributed into pairs by the TA and the instructor. This was in accordance with the experiments presented by Khan[6] and Mendes[11]. The factors that were taken into consideration when distributing the pairs their preference in working with a partner of the same or opposite gender, and the partner's academic level. The aim was to try to find the most

compatible pairs according to the students' preferences, as indicated in the initial questionnaire.

3.2 Data Collection

The data collected throughout the semester was in the form of questionnaires, video recordings, course work assessment, and observations done by the instructor and TA during the labs.

3.2.1 Initial Questionnaire

Before starting the experiments, a questionnaire was designed and distributed among the students of both the PP and the traditional sections. The questionnaire was designed after studying several similar questionnaires that were designed for similar experiments[4, 14, 47], while taking into consideration the background and mentality that distinguished our students from those in other countries.

The questionnaire aimed to give us a general idea of the students' academic background, and their preference as to working in groups. The questionnaire also asked about the students' partner preference from an academic aspect, as well as their gender. The full questionnaire is under Appendix III.

3.2.2 Videos/Pictures/In-lab observations

The in-class performance was observed and studied through the videos and photos that were recorded during the lab. A 10-minute video was recorded after the first

half hour of each lab. From these videos, we tried to study the effects that PP had on how the students were performing and how they were behaving within the lab.

Once the pairs were formed, video recordings were taken every lab, for both sections. The videos were 10 minutes long on average, and were recorded after the first half hour of the lab was over. The reason that the video was recorded after the first half hour of the lab because that time is usually allocated for the instructor to explain the experiment, and for the students to prepare for the session. After the first half hour is over, students are usually occupied with programming, and the videos would capture the effect of PP more accurately. The length of the video was selected to be long enough to allow the measure of the parameters, but not too long as to make the analysis process overly time consuming. Pictures were also taken during both labs.

We were limited to using one video camera for the duration of the experiments, and therefore, were limited to focusing on one pair, or two individual students in every video recording. Also, it limited us to focusing either on the students themselves, or on the screen in front of them. We decided to focus on the students and the keyboard they were using, as they gave us the most data in our setup.

In addition to the video recording and the pictures, notes and observations were made both by the instructor and the TA during the labs in both sections. These observations pertained to the interactions between the students, the number and type of questions asked, the time required to complete tasks, and the degree of enjoyment of the labs session. In addition, in the lab sessions of the PP section, it

was observed how often students switched roles, how helpful and attentive were the navigators, and to what extent the tasks were discussed among each pair.

3.2.3 Code Analysis

Code samples were collected from both the PP and the traditional sections at the end of the second semester. By that time, the students were supposed to be able to write and entire program, with a proper user interface. This code was analyzed for quality, length, structure, and complexity. The code was graded, with the grade considered as a measure of correctness, but was not included in the students' final grade.

The program that the students were asked to write was a simple memory game. The goal of the game was to find every pair of matching card images of eight pairs on a 4X4 grid in as few tries as possible. The program was supposed to display the images, and flip one at a time when the player clicked on it using a mouse. Only two cards were turned over at a turn, and were supposed to flip back unless they matched. The interface include a turn counter, of the number of times two cards were flipped, and a new game, and an exit button.

3.2.4 Work Assessment

Even though the students were encouraged to work together and help one another, the quizzes and assignments were still individual work, and any collaboration was

prohibited. During the semester, the students take four quizzes and submit four assignments.

In addition, a midterm exam, a practical final and a written final exam are taken into consideration. The midterm exam took place during the semester, and the practical and written finals are scheduled at the end of the semester.

Finally, the drop rate and the attendance and average absence from lectures and labs are taken into consideration.

3.2.5 Follow-up Questionnaire

A follow up questionnaire is designed to measure the response of students to PP. Similarly to the initial questionnaire, a number of papers that demonstrated similar experiments[4, 7, 8] were consulted, in order to detect the most relevant parameters that this questionnaire can measure.

The questionnaire focuses on how useful students found working with PP, how easy the students found to work with a partner, and their thoughts on the effect of applying PP on in-class tasks in comparison to the traditional methods of teaching. The complete questionnaire is under Appendix IV.

3.3 Data Analysis

The analysis of the data collected differed according to the type of data, and the tool that was used for this analysis.

3.3.1 Questionnaire Analysis

Since the initial and follow-up questionnaires did not contain complicated questions that needed elaborate statistical analysis, the students' answers were transferred to a Microsoft Excel © spreadsheet, and the relevant statistics, such as the averages, and count values were calculated.

3.3.2 Video Analysis

These videos were analyzed using ELAN1. ELAN is a software tool developed in The Language Archive, in the Max Planck Institute for Psycholinguistics, which allows annotations to be added to videos. These annotation were used in the context of our experiment to determine the amount of time that students spent talking, typing, gazing off, asking questions, and laughing. After these numbers were retrieved, they were further analyzed using the appropriate statistical hypothesis tests, discussed in the following section.

The reason for selecting ELAN for the annotation, was that in addition to the ability to create multiple tiers to annotate different behaviours in a video[48], there are few restrictions on the annotation layers[49]. This allowed us to create tiers corresponding to relevant behaviours in our study, and to annotate each of the tiers separately from the others in each video segment.

[1]Available from: https://tla.mpi.nl/tools/tla-tools/elan/.

3.3.3 Student Code Analysis

A sample of the students' code was collected during the second semester experiment. The code was analyzed using SourceMonitor[2]. According to Alemerien and Magel in their research[50], SourceMonitor has proven to be an efficient tool, and the results it produce are the closes to manual analysis result when compared to other tools. This software offers an analysis of code in a number of programming languages, including java. The statistics it gives pertain to the number of lines, classes, methods, and the percentage of comments. In addition, it offers statistics about the maximum and average depth and complexity of the code.

The parameters that were used were selected by reviewing the ISO/IEC 9126 standards for code quality, as well as the analyses that are given by SourceMonitor. ISO/IEC 9126 states that a source code quality depends on its analyzability, changeability, stability, and testability[51].

The analyzability of a code is defined as "the capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified."[51] This can be measured by the number of statements and comments.

[2] Available from: http://www.campwoodsw.com/sourcemonitor.html

The code changeability is its ability to accommodate the implementation of modifications. This is done through modularity, which is measured by the number of methods in the code.

The code is considered to be stable if it is able to avoid any "unexpected effects from modifications of the software."[51] One of the measures of code stability is the number of calls that are made in the code.

Finally, the testability of the code is its ability to enable the validation of the software after any modifications are made. This can be measured by the number of non-cyclic paths as well as the number of nested levels in the code, which are measured by the code complexity and depth respectively.

3.3.4 Statistical Hypothesis Tests

A statistical hypothesis was defined by Wyllys as *"a statement about the value of a population parameter (e.g. mean, median, mode, variance, standard deviation, proportion, total)"* [52]. The t-test was selected to study our data because it is applicable when we have two independent samples, with equal means and population sizes[53]. The test was applied separately on the each of the experiments, to measure the relevance of the difference in results between the PP section and the traditional section.

The t-value is calculated according to equation 1.

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_{x_1 x_2} \cdot \sqrt{\frac{1}{n}}} \quad (1)$$

Where

$s_{x_1 x_2}$ is the grand standard deviation, as in equation 2.

$$s_{x_1 x_2} = \sqrt{\left(s_{x_1}^2 + s_{x_2}^2\right)} \quad (2)$$

$s_{x_1}^2$ is the unbiased estimator of the variance of group 1

$s_{x_2}^2$ is the unbiased estimator of the variance of group 2

Microsoft Excel© offers a function that calculate the t-values for any set of data, and this function was used to calculate the t-values for this experiment. The function that was used was part of the Excel Analysis ToolPak add-in, shown in Figure 15.

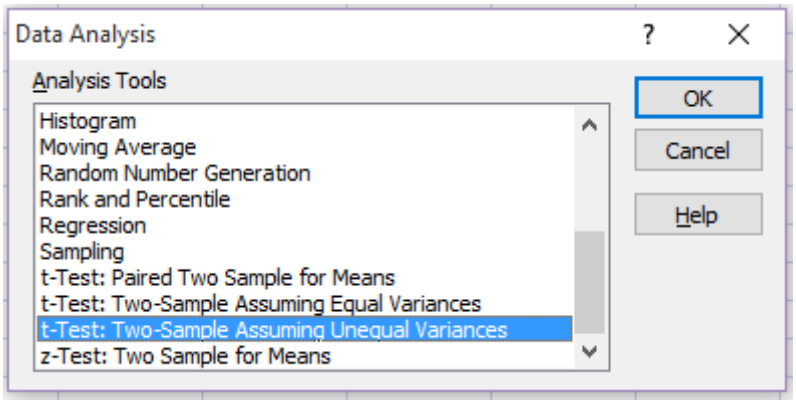


Figure 15 - A Snapshot of the Excel Analysis ToolPak

After calculating the variances of the data that was obtain, it was determined that the appropriate test to use was the t-Test: Two-Sample Assuming Unequal Variance. The tool gives the ability to select the cell ranges of the variables that are being tested as seen in Figure 16.

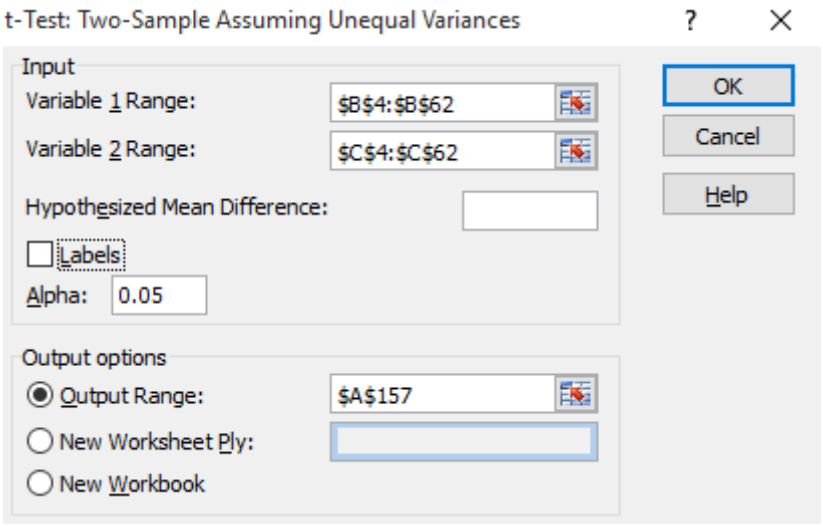


Figure 16 - A Snapshot of the Two-Sample t-Test Assuming Unequal Variances

When the value of the t-test is calculated, the result is shown in the sheet, in the format illustrated in Figure 17. The result includes calculating the mean and the variance of the two data sets, determining the degree of freedom (df), and the value of the t-test (t Stat). The difference is considered to be significant if the absolute value of t Stat is larger than the value of the t Critical one tail.

t-Test: Two-Sample Assuming Unequal Variances

	Variable 1	*Variable 2*
Mean	6.526786	4.909091
Variance	6.267451	5.158249
Observations	56	55
Hypothesized Mean	0	
df	108	
t Stat	3.566761	
P(T<=t) one-tail	0.00027	
t Critical one-tail	1.659085	
P(T<=t) two-tail	0.00054	
t Critical two-tail	1.982173	

Figure 17 – A Snapshot of the Output of the t-Test

In the above example, we can assume that the difference between the results of Variable 1 and Variable 2 are significant, where Variable 1 got better scores. This assumption comes as the result of having the t-test value larger than the t Critical one tail. This is denoted by $T(108) = 3.566761$, $p < 0.00027$.

Chapter 4 Results and Discussion

This chapter presents the results that have been reached by this research, from the field experiment in the first section. It goes on to discuss the results in the second section. The presented results include the results that were collected over the two phases of the experiment during the academic year 2014 – 2015.

4.1 Results

This section presents the results of the data that was collected and analyzed during the two iterations of the experiment. This includes data collected from the two questionnaires given to the students, the analysis of the video recorded during the labs, and the code collected from students, as well as the grades of the students.

4.1.1 Initial Questionnaire

The initial questionnaire that was given to the students at the beginning of the semester aimed to collect some demographic information about the students, their academic level, and their preferences to working within a team. The questionnaire was designed at the beginning of the first semester.

Figure 18 illustrates the distribution of students according to gender in both sections. It shows that in the PP sections, the students were distributed evenly, with 50% males to 50% females. In the traditional section, 55% were males, while 45% were female.

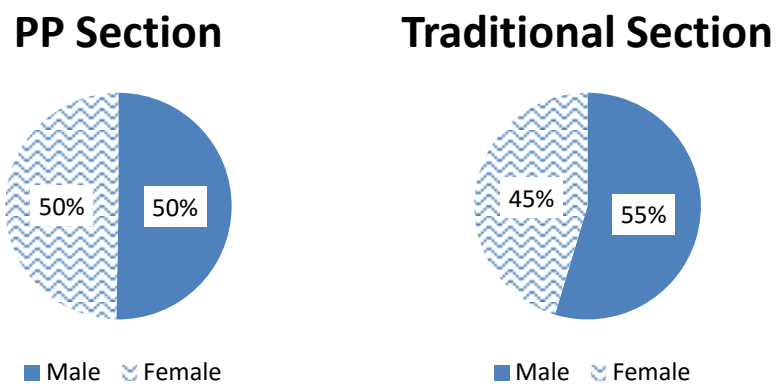


Figure 18 - The Distribution of Students according to Gender

According to the study plan for CS and CSE students, Comp231 is a second-year course, making most students who register for it sophomores. Nevertheless, students who fail the introductory course preceding Comp231, and the Comp231 itself, may be required to take it during their junior year. In addition, students from specializations other than CS and CSE, who take it as an elective or a requirement to a minor degree register for the course later than during their second year of university.

Figure 19 shows that the students were distributed almost evenly. 77% of the PP section being sophomores, and 23% being juniors or seniors. Similarly, 78% of the students in the traditional section were sophomores, and 22% were juniors or seniors.

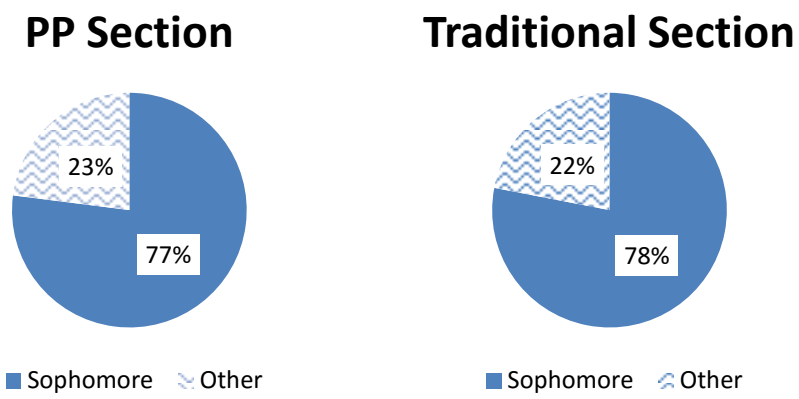


Figure 19 - The Distribution of Students according to University Level

Since almost a quarter of the students were in a higher level than sophomores, we wanted to know if there were students who have registered for the course previously. This could mean that they have either failed the course, dropped it, or did not get a good enough grade, which require them to register for it again. When asked if this was the first time they register for the Comp231 course, 65% of the PP section students answered yes, and 35% answered no. As for the traditional section, 73% indicated that they are registering for the Comp231 for the first time, and only 27% have previously registered for the course. These percentages are indicated in Figure 20.

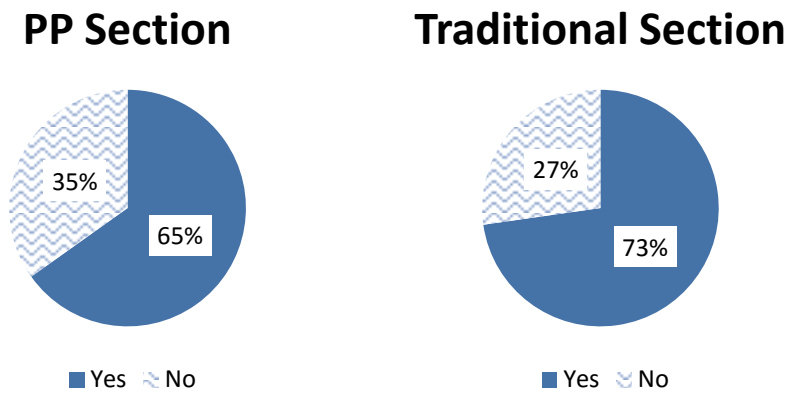


Figure 20 - Students Distribution according to their Registering for the Comp231 Course for the First Time

The other aspect that the questionnaire was designed to shed light on was the students' preferences when it comes to working in teams. As expected, very few students in both semesters indicated that they preferred to work individually, with 17% in the PP section, and 20% in the traditional section. The details of the distribution are shown in Figure 21.

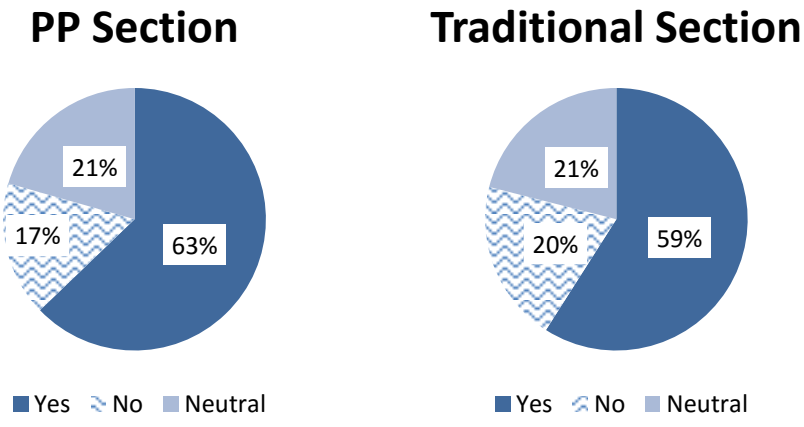


Figure 21 - Students Distribution according to Preferring to Work in Pairs

In addition to that, the vast majority of the students preferred to select their own partner, as indicated in Figure 22. In the PP section, 87% indicated that they preferred to select their partner, as did 93% of the traditional section.

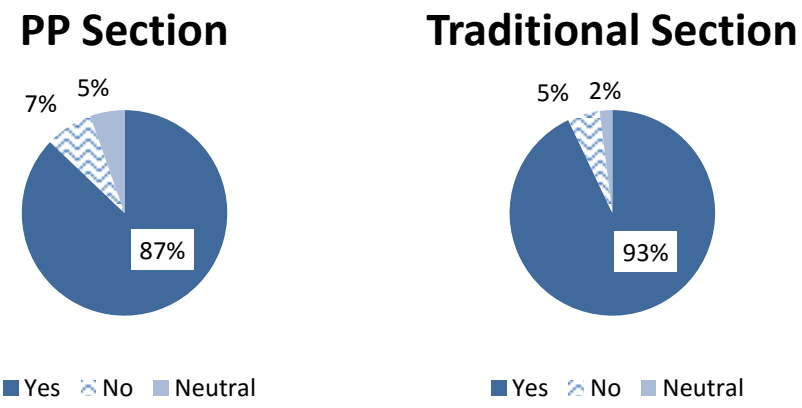


Figure 22 - Students Distribution according to Preferring to Select Their Partner

When it came to the partner gender or specializations, shown in Figures 23 and 24 respectively, most students did not have an issue with working with partners from either gender, and from whichever specialization.

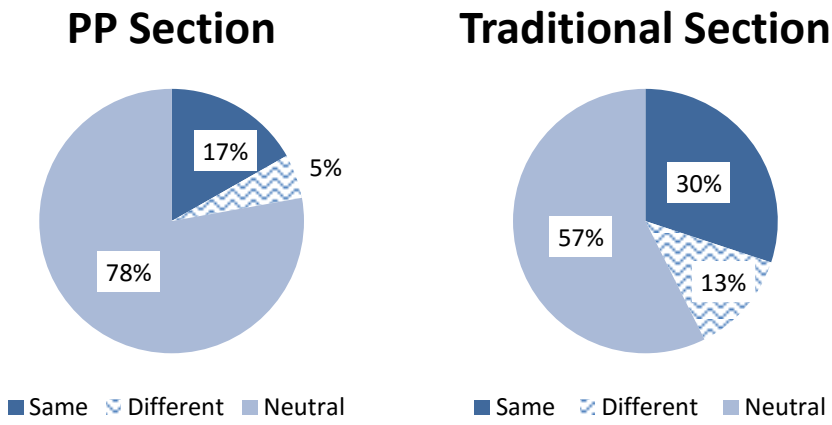


Figure 23 - Students Distribution according to Partner Gender Preference

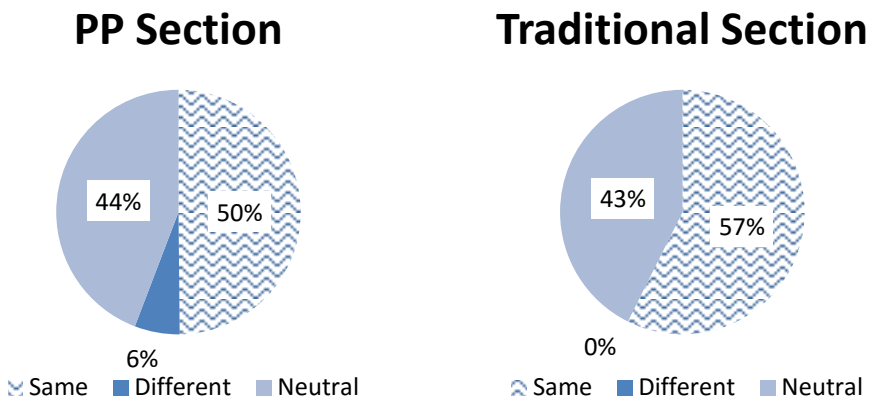


Figure 24 - Students Distribution according to Partner Specialization Preference

However, most students indicated that they would prefer to work with a partner with the same programming level as they are, as shown in Figure 25. 53% of the PP section students indicated that they would prefer to work with a partner in the same programming level as they were, and 23% of the students indicated wanting to work with a partner in a different programming level, whether it be higher or lower. In the traditional section, 56% of the students indicated wanting to work with a partner at their programming level, and 23% of them preferred to work with a partner with a different programming level than theirs.

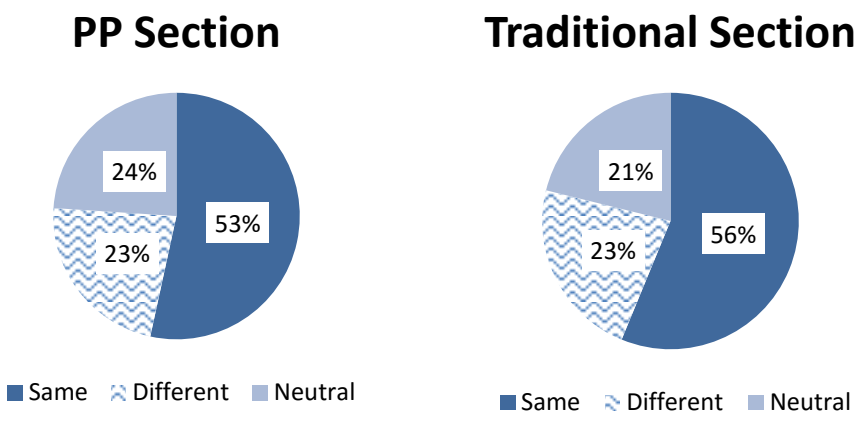


Figure 25 - Students Distribution according to Partner Programming Level Preference

Despite their answers, it was noted that when students selected their own partner, the partner's programming level did not play a role in the process, as will be further discussed.

4.1.3 In-class Performance

The analysis of the videos focused on the amount of talking and typing that the students did. In addition, it took notice of the amount of gaze-off and smiling that took place. Finally, the number of times they asked for help from one of the instructors, and pointed to the screen were recorded.

The video analysis was done using the ELAN software, shown in Figures 26 and 27. For every video, six tiers were defined; Gaze-off, Typing, Talking, Smiling, Asking for Help, and Pointing at Screen. Each of these tiers was a parameter that was being measured. The output contained the length of each of the intervals indicated with the software. The percentages were calculated from the output.

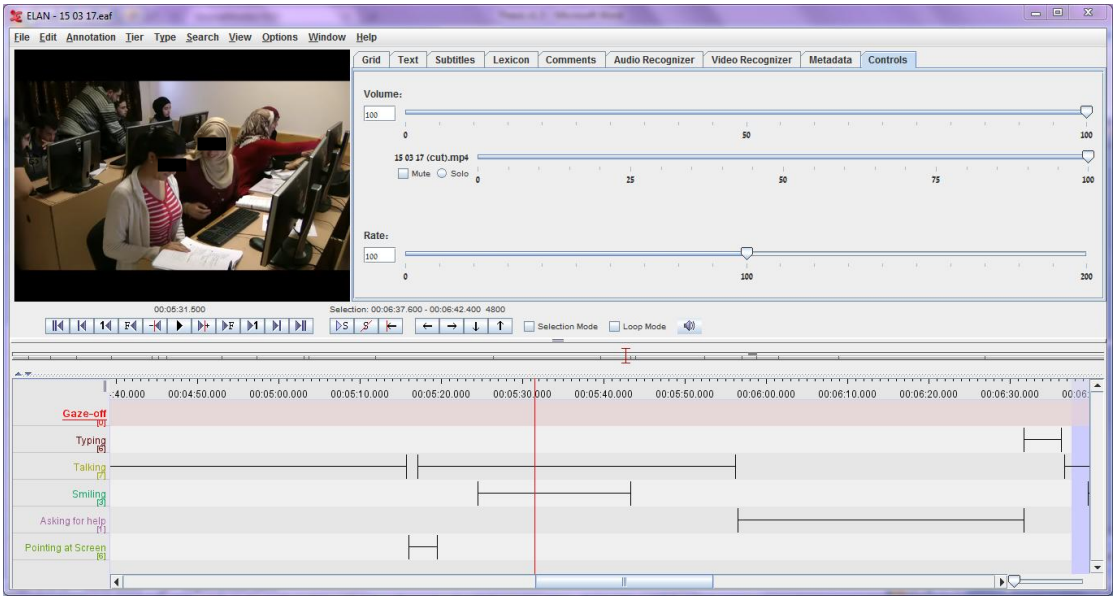


Figure 26 - A Snapshot of the Video Analysis using ELAN from the PP Section

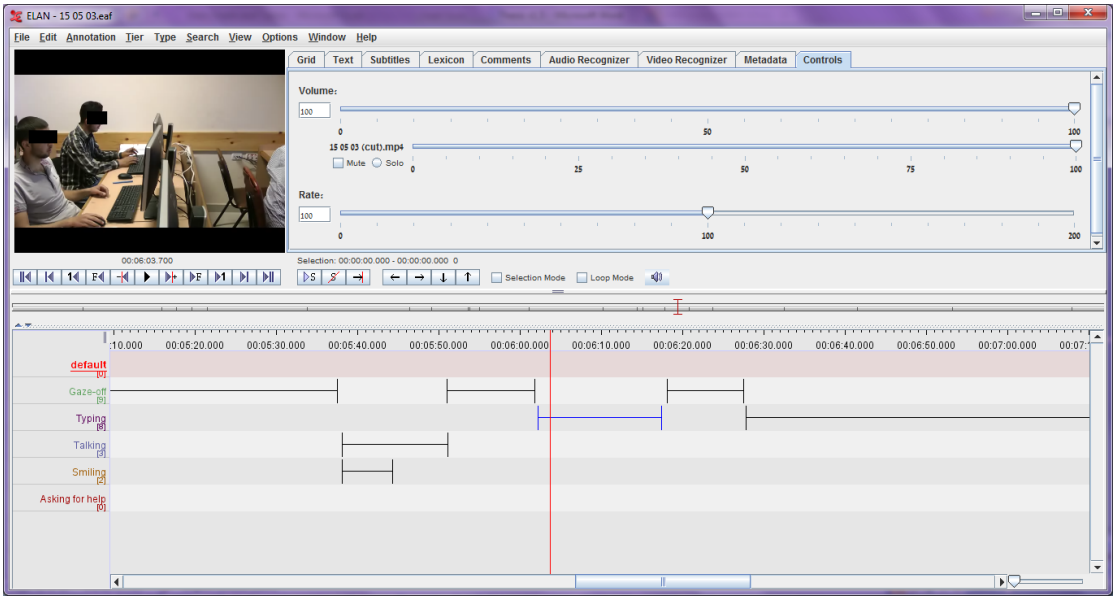


Figure 27 - A Snapshot of the Video Analysis using ELAN from the Traditional Section

Figure 28 shows that the time that students spent typing in the traditional section was significantly less than their peers in the PP section ($T(4) = -1.26278$, $p <$

0.15). In addition, time spent at gaze-off was not spotted in the PP section, but made up 17% of the time in the PP section ($T(2) = -4.06229$, $p < 0.001$).

On the other hand, the time spent talking by the students in the PP section was significantly higher than in the traditional section ($T(11) = 7.561836$, $p < 0.001$). In the PP section, talking time overlapped typing, smiling, asking for help, and pointing to screen. The students in the PP section also smiled significantly more than the students in the traditional section ($T(6) = 2.080897$, $p < 0.05$).

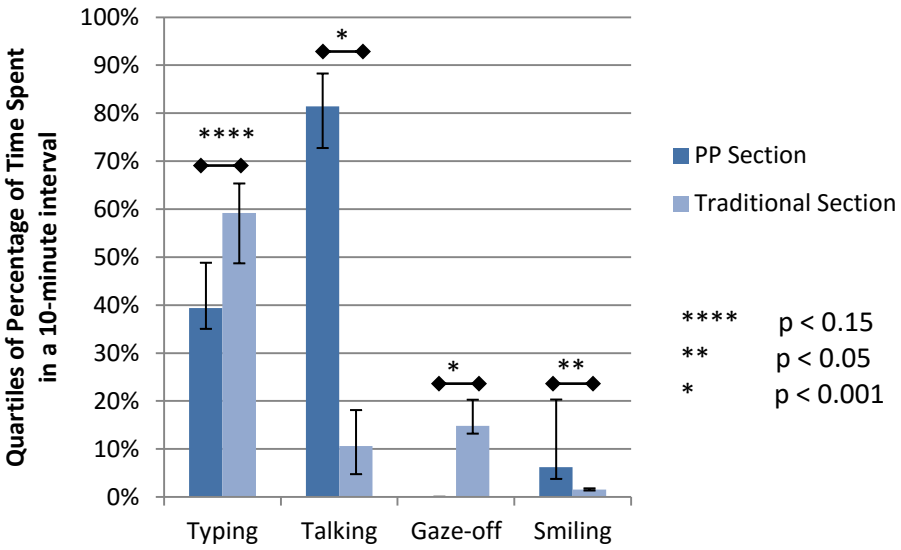


Figure 28 - Students Distribution according to Time Distribution from the Video Analysis

Figure 29 shows that students in both sections asked for help, without their being a significant difference between the sections ($T(3) = -0.70857$). However the students in the traditional section were never recorded pointing to the screen ($T(7) = 6.386948$, $p < 0.001$).

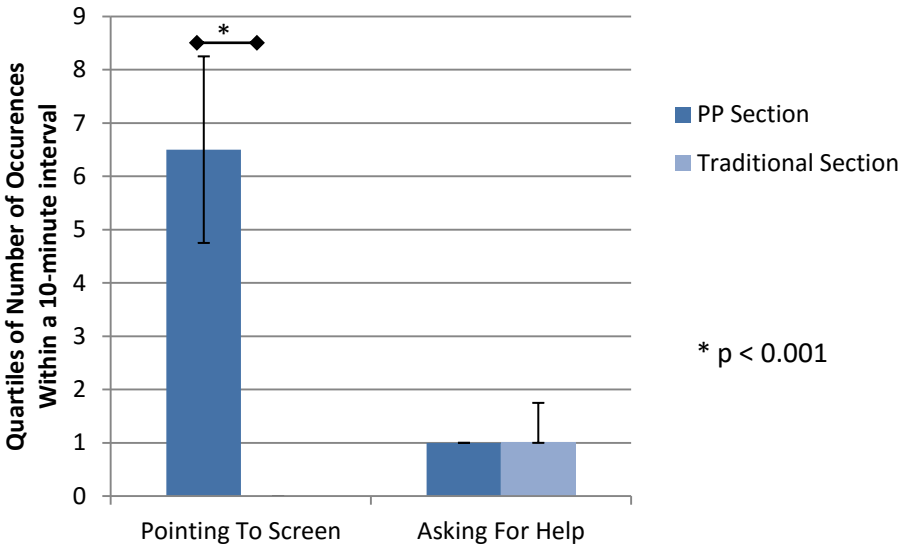
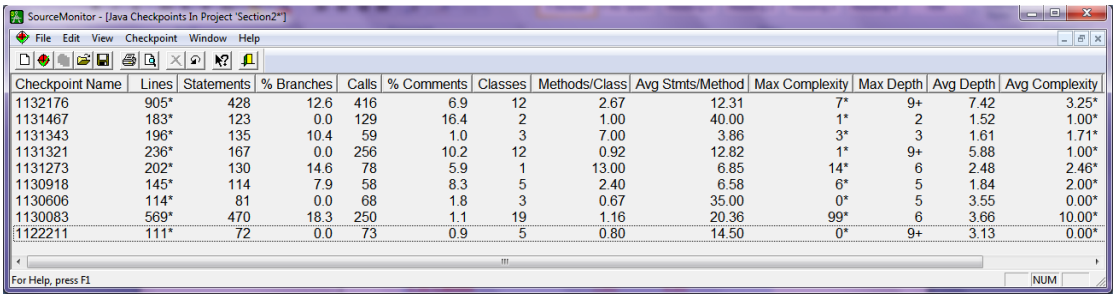


Figure 29 - Students Distribution according to Behaviour from the Video Analysis

4.1.4 Code Quality

To understand the effect that PP had on the quality of the code that was produced by the students, a sample of the students' code was collected and analyzed using SourceMonitor, as shown in Figure 30.



Checkpoint Name	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Stmts/Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity
1132176	905*	428	12.6	416	6.9	12	2.67	12.31	7*	9+	7.42	3.25*
1131467	183*	123	0.0	129	16.4	2	1.00	40.00	1*	2	1.52	1.00*
1131343	196*	135	10.4	59	1.0	3	7.00	3.86	3*	3	1.61	1.71*
1131321	236*	167	0.0	256	10.2	12	0.92	12.82	1*	9+	5.88	1.00*
1131273	202*	130	14.6	78	5.9	1	13.00	6.85	14*	6	2.48	2.46*
1130918	145*	114	7.9	58	8.3	5	2.40	6.58	6*	5	1.84	2.00*
1130606	114*	81	0.0	68	1.8	3	0.67	35.00	0*	5	3.55	0.00*
1130083	569*	470	18.3	250	1.1	19	1.16	20.36	99*	6	3.66	10.00*
1122211	111*	72	0.0	73	0.9	5	0.80	14.50	0*	9+	3.13	0.00*

Figure 30 - A Snapshot of Code Analysis using SourceMonitor

Nine programs were collected from each of the PP and traditional sections, and run through the software. The code was for a game of matching each two identical

cards on a board of 16 overturned cards. The students had to write the software, and design a graphical user interface (GUI) to go with it.

SourceMonitor gives statistics about a number of parameter, per program. It counts the number of lines, statements, calls, and classes in the code. In addition, it gives the percentage of branches and comments. Moreover, it offers the ration of methods to classes, and statements to methods. Finally, it calculates the average and maximum complexity and depth.

In addition to the statistics produced by SourceMonitor, the number of methods in each program was counter, as well as the number of syntax errors the software had.

Figure 31 shows that the code written by PP students is shorter on average, by about a third ($T(9) = -1.13395$, $p < 0.15$), with fewer statements ($T(10) = -1.10604$, $p < 0.15$), and the calls ($T(11) = -1.11517$, $p < 0.15$).

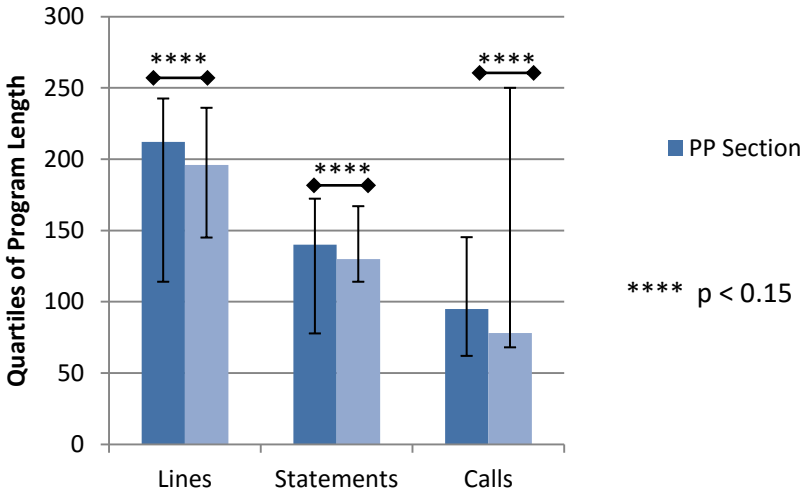


Figure 31 - Code Statistics from SourceMonitor Regarding Program Length

As demonstrated in Figure 32, the PP section's students wrote code that contains more classes (T(15) = 0.26165). However, The number of methods in the traditional section's code was more than those in the PP section (T(15) = -0.39541). Nevertheless, neither difference was significant.

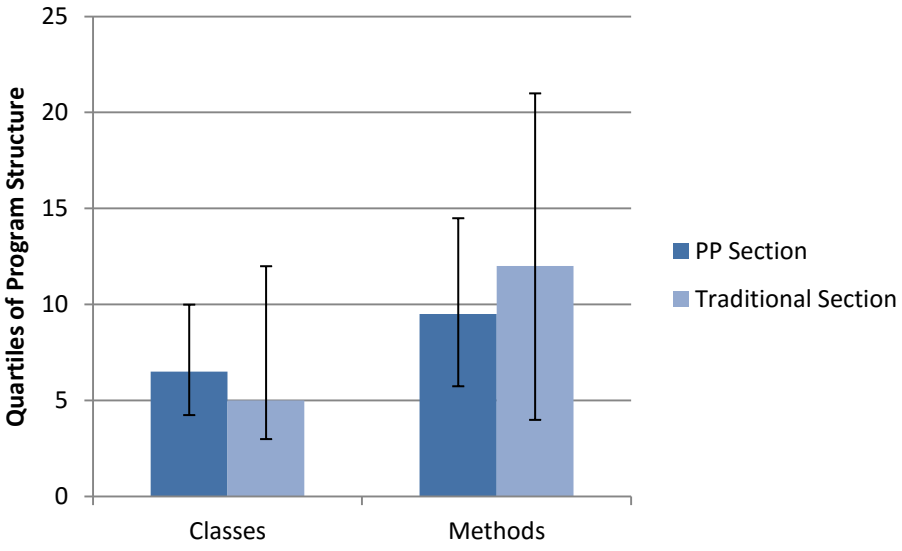


Figure 32 - Code Statistics from SourceMonitor Regarding Program Structure

This resulted in significantly less errors (T(8) = -1.13281, $p < 0.15$), as shown in Figure 33.

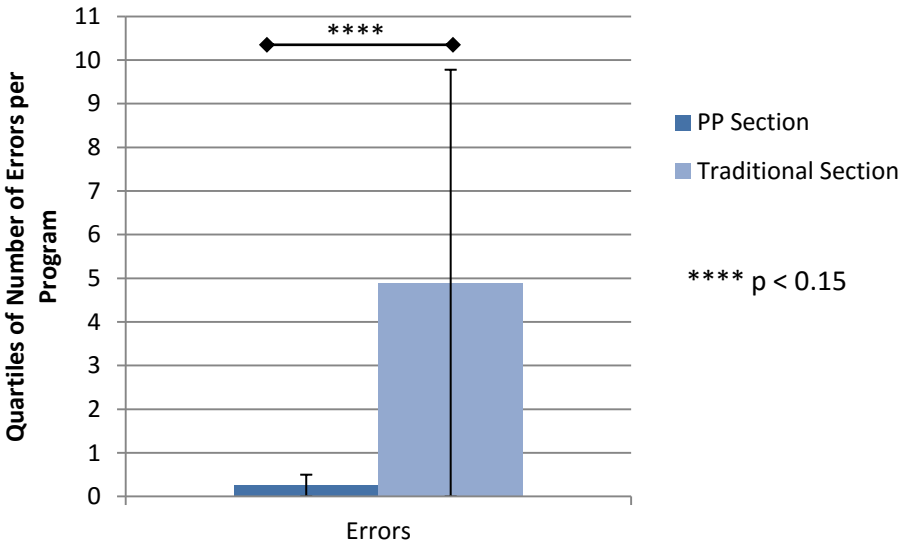


Figure 33 - Number of Errors per Program

Despite being shorter, the code written by students in the PP sections was better commented that the code written by the traditional section's students ($T(15) = 1.736667$, $p < 0.1$), as shown in Figure 34.

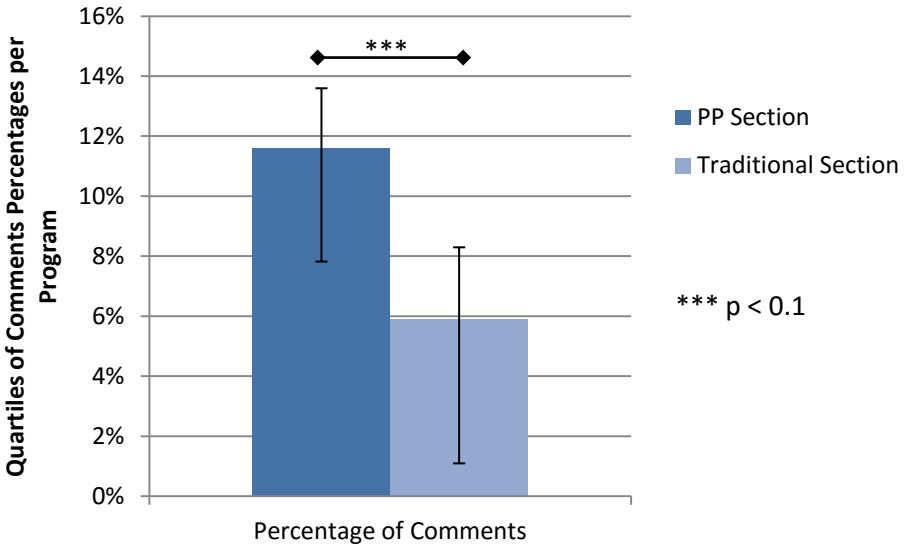


Figure 34 - The Percentage of Comments in the Students Code according to SourceMonitor

SourceMonitor defines average block depth as *"the average nested block depth weighted by depth"*[54]. Figure 35 shows that there was not a significance difference in the average depth of the code written by students in the PP section as opposed to students in the traditional section (T(15) = -0.50252). However, when it comes to complexity, which is defined as *"the overall complexity measured for each method (and, if present, each function) in a file or checkpoint"*[54], the code written by the students in the PP section was significantly more complex, at an 80% confidence, than that written by their peers in the traditional section (T(13) = -1.01563, $p < 0.2$).

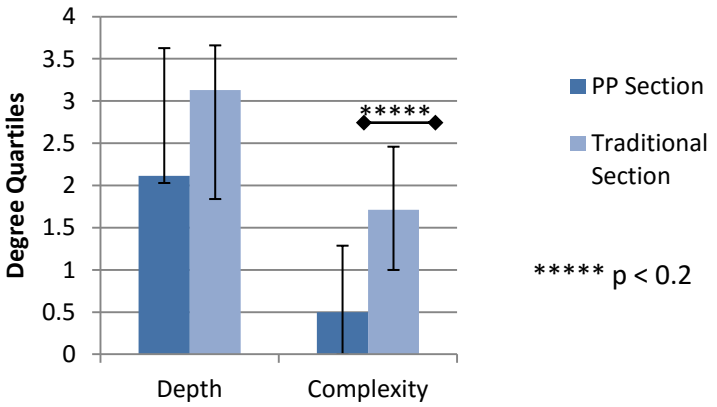


Figure 35 - The Depth and Complexity of Students Code from SourceMonitor

Despite those differences, the grades that the students in both sections received were not significantly difference (T(15) = 0.249932), as can be seen in Figure 36.

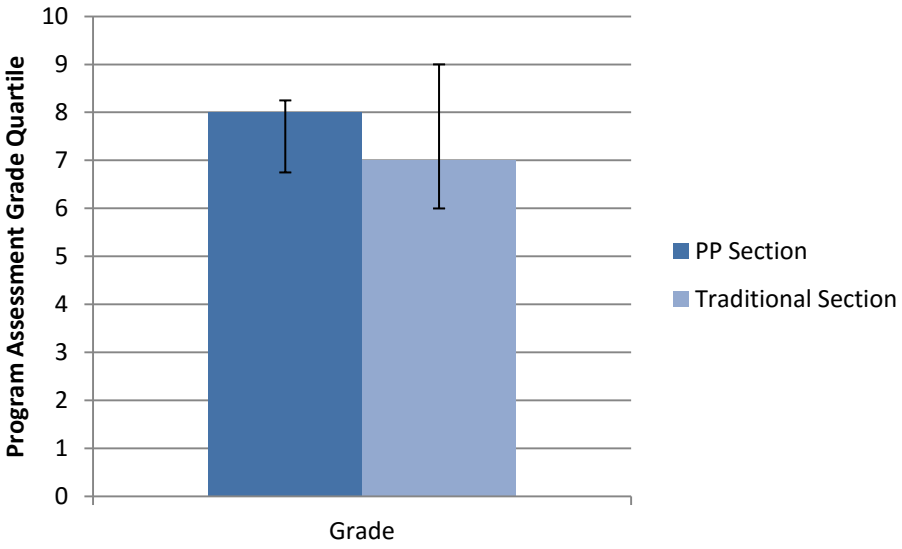


Figure 36 - Code's Assessment Grade

4.1.5 Course Assessment

The course assessment was measured firstly through the grades that the students received during the semester, and secondly through the drop and completion rates of the students in the course. The grades were the results of four quizzes, four assignments, a practical exam, a midterm and a final.

The quizzes that the students took were always of the same complexity and difficulty levels, but never the same. The quizzes were given during the lab, and often contained a written part, and a programming part. They were given either at the beginning of the lab session, or at the end of it, and were usually assigned twenty minutes to be solved.

From Figure 37, it can be noted that the PP section's grades were significantly better in the first ($T(108) = 3.566761$, $p < 0.001$) and third ($T(94) = 1.556654$, $p <$

0.15) quizzes, as well as in the quizzes total grade (T(102) = 3.111714, p < 0.001), but not in the second (T(94) = -0.14678) and fourth (T(66) = 1.370113) quizzes.

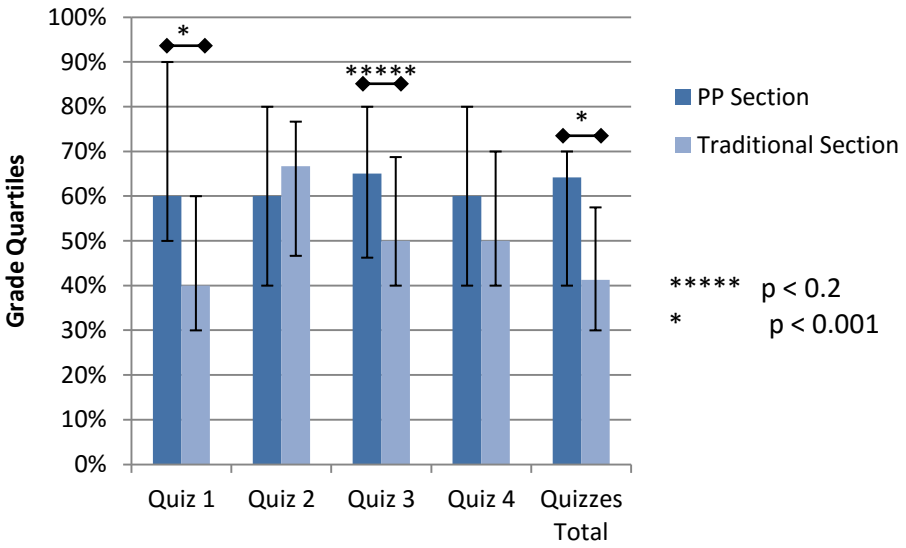


Figure 37 - Students Performance in Quizzes

During each semester, four assignments were given to the students. The assignments increased in difficulty as the semester progressed, but always included concepts that were fully covered during the lecture and lab sessions. Students were asked to work individually on these assignments, and any cheating cases that were punished. The assignments were the same for all sections, and were distributed at the same time. Each assignment was given two weeks to be completed and turned in.

Figure 38 shows that the only significant difference in assignment grades was in the third assignment (T(86) = 1.53477, p < 0.2), and the assignments total (T(98) = 1.699682, p < 0.1). However, in the first (T(98) = 1.021331), second (T(88) = -

0.21876), and fourth (T(50) = 1.266833) assignments, the differences were not of significance.

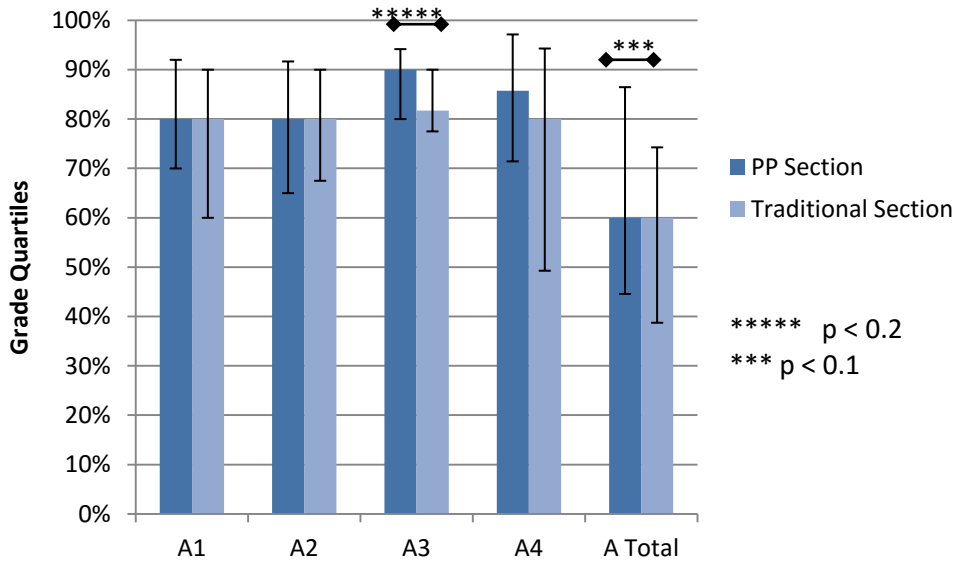


Figure 38 - Students Performance in Assignments

At the end of the semester, a practical exam is given to the students. This exam is a programming exercise that covers all the programming concepts that were introduced throughout the semester. Both sections are given the same exam, and it is usually allotted 90 minutes. Figure 39 shows that the difference between the performance of the students of the PP section and the traditional section was not significance (T(81) = 0.57024).

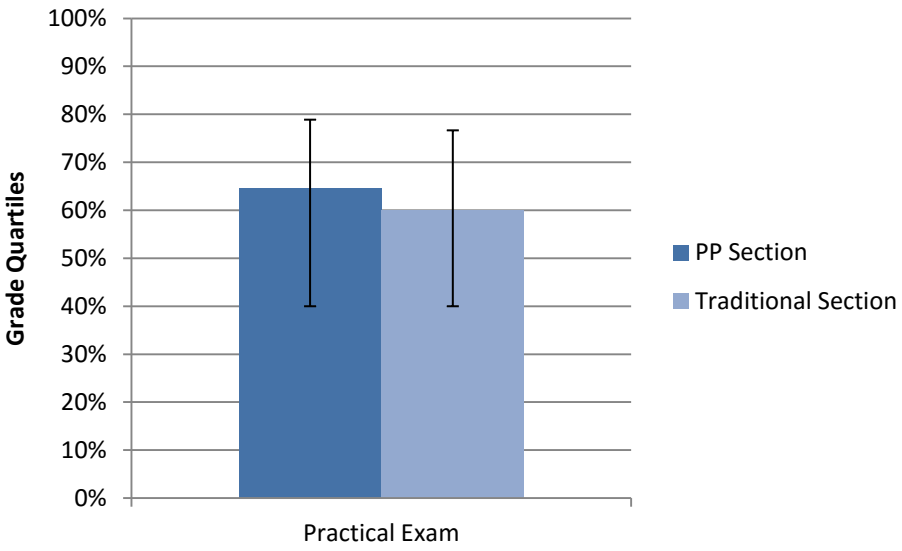


Figure 39 - Students Performance in the Practical Exam

The students performance in the quizzes, assignments, and the practical exam is combined to create their lab total. The lab work has a weight of 35% of their semester work. Figure 40 shows that the students in the PP section perform significantly better, when the lab total is calculated, ($T(101) = 2.429545$, $p < 0.001$).

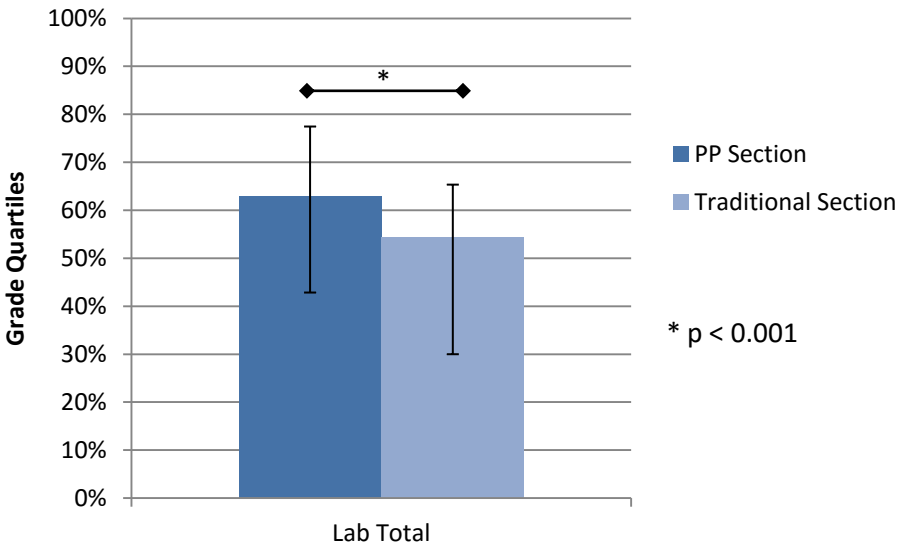


Figure 40 - Students Performance in the Lab

The written exams during the semester are distributed on a midterm and a final. Both sections take the same exams at the same time. The students in the PP section performed better in the midterm than the students in the traditional section ($T(95) = 2.510584$, $p < 0.001$). However, in the final, the difference between both sections was not significant ($T(87) = -0.37894$).

The final student grade is made up from the lab total, and the midterm and final exams. Figure 41 shows that the PP section performed significantly better in the overall result of the course ($T(88) = 1.612096$, $p < 0.15$).

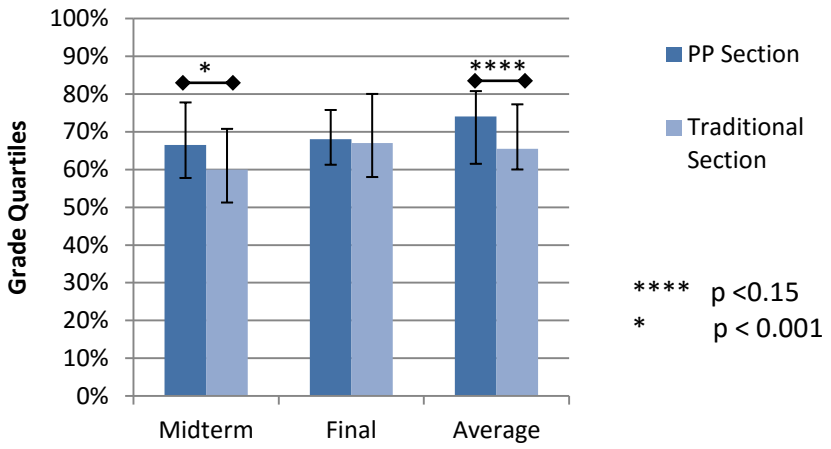


Figure 41 - Studnets Performance in Written Exams and Final Average

The drop rate and absence rate in both PP sections was less than the half of the drop rate in the traditional sections, as shown in Table 2. However, this affected the fail rate, making it a little higher in the PP section (17%) than in the traditional section (14%). More students tended to miss class in the traditional section, with absences averaging near four students per class, where in the PP section this rate was a little higher than two students per class.

Table 2 - The Students Drop and Fail Rates, and Absence Average Per Semester

	PP Section	Traditional Section
Drop Rate	15%	32%
Fail Rate	17%	14%
Absence Average	2.26	3.85

4.1.6 Follow-up Questionnaire

The follow-up questionnaire was designed to measure how much the students enjoyed working in pairs, and how useful they felt this teaching method was. The

same questionnaire was only given to the PP sections in both semesters, and was distributed at the very end of the semester. The full questionnaire is found under Appendix IV.

The statements relating to the PP technique, presented in Figure 42 are:

- Q1) PP helped me in better understanding concepts that were ambiguous during the lecture.
- Q3) PP helped me to solve lab problems faster.
- Q4) PP helped me to solve individual assignments faster.
- Q5) PP made programming more fun.
- Q11) PP made my overall learning experience better.

Most of the answers to the questions asked in the questionnaire were favourable towards PP, as shown in Figure 42. 82% of all students thought that PP helped them in better understanding concepts that were unclear or confusing to them during the lectures. When it comes to improving their problem solving skills, 73% thought that PP helped them solve the lab problems faster, and 55% thought that this effect extended to include individual homework and assignments. PP made programming more fun for 88% of the students, while 82% of them thought that PP improved their overall learning experience in this course.

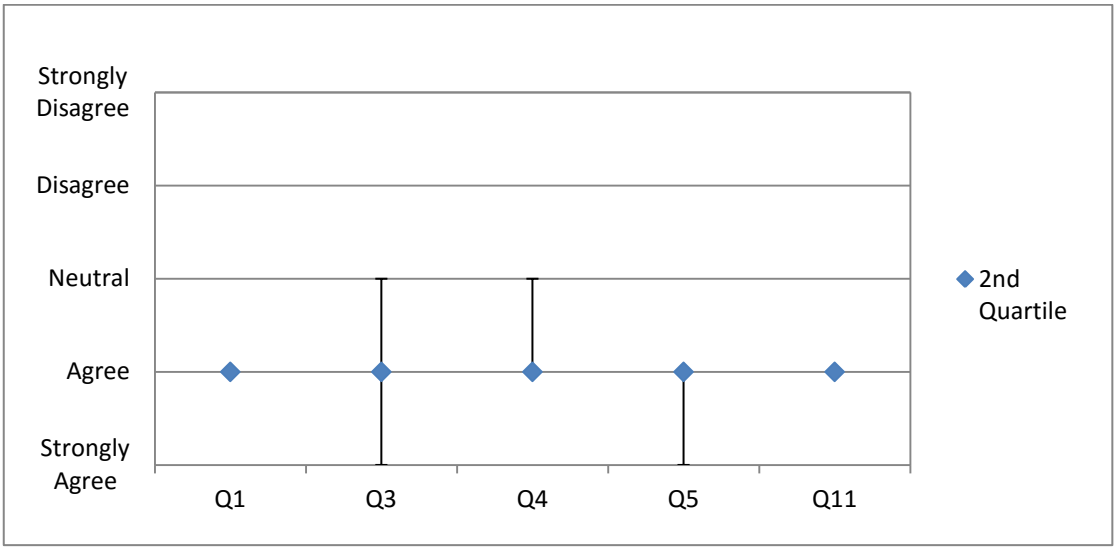


Figure 42 - Follow-up Questionnaire Answers Regarding PP

However, when the questions were directed toward the pair dynamics, some of the issues with PP might be detected. Although most of the answers were still favourable towards the partners, they were not as conclusively positive as the answers that were directed towards PP itself.

The statements about the pair dynamics that were in the questionnaire are the following:

- Q2) I was able to learn new concepts from my partner
- Q6) I got along well with my partner.
- Q7) My partner and I switched roles between driver and navigator regularly.
- Q8) My partner and I distributed the work equally.
- Q10) My partner and I were of similar academic level.

Although only 37% of the students felt that they were of the same academic level as their partners, as shown in Figure 43, 71% of all students thought that they

learnt new concepts from their partner. 82% thought that there was a mutual understanding between their partners and themselves. When it came to the distribution of tasks and work, 71% claim that they switched roles between driver and navigator regularly, and 69% believe that they distributed the tasks among themselves equally.

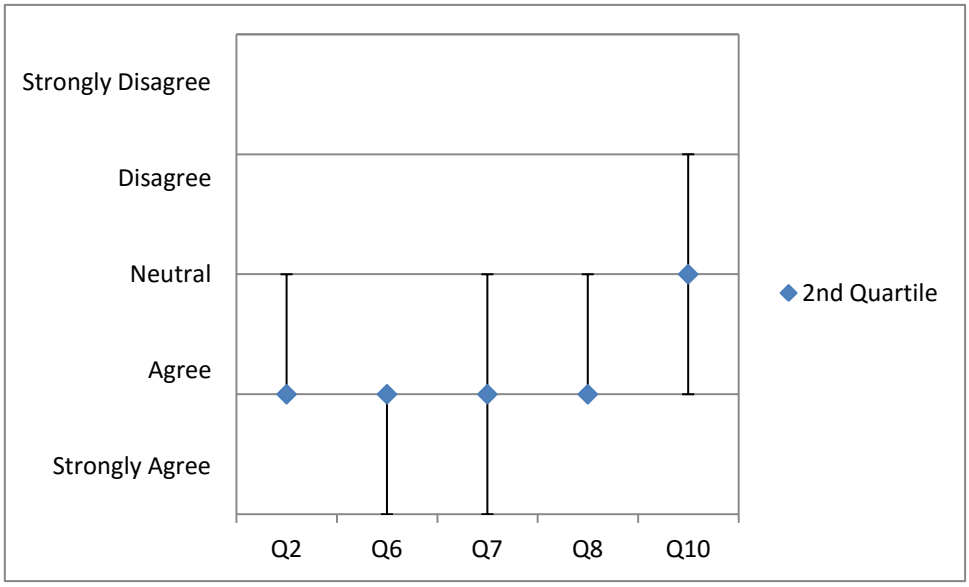


Figure 43 - Follow-up Questionnaire Answers Regarding Partner

Regarding their roles, 49% of the students felt that they benefited more out of the exercises when they were acting as a driver, as illustrated in Figure 44. On the other hand 18% felt that the benefit was more when they assumed the navigator role. 33% of the students didn't feel strongly towards one role or the other, which is the ideal situation in PP.

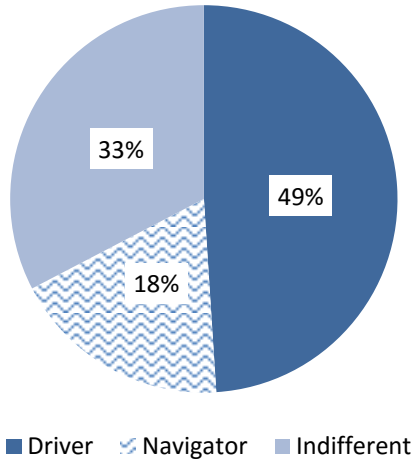


Figure 44 - Students Preference to PP Roles

Finally, when asked if they would like to learn other courses through PP, 82% of the students agreed that they would, as shown in Figure 45. Only 8% said that they would not like to have PP principles applied to another course they study in the feature.

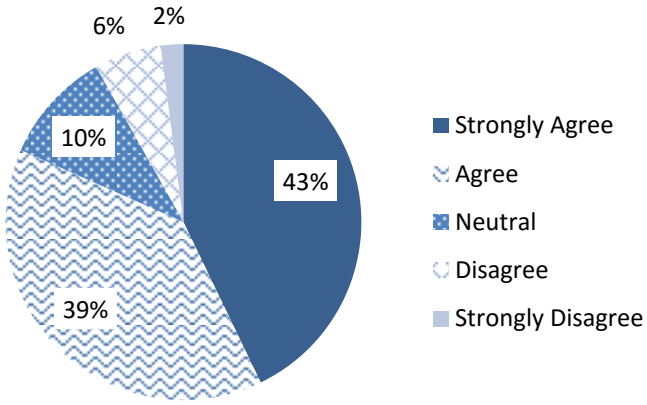


Figure 45 - Students Willingness to Use PP in Other Courses

4.2 Discussion

This section discusses the results that were presented in the previous section. This discussion attempts to put the data that was produced by the experiment into the context of the experiment, and the society. In addition, it offers the insights and experience of the researcher regarding the results.

4.2.1 Initial Questionnaire

One of the key variables that were measured by the initial questionnaire was whether the students were registering in the course for the first time or not. Even though the initial inclination would be towards thinking that students who are taking the course for the second time will have prior knowledge about the topics being presented, which gives them an advantage.

Nevertheless, during the years prior to this experiment, it was observed by the researcher that students who fail the course or drop out of it, and are forced to take it again rarely show any improvement during their second time around. This is because most failure or drop cases are due to the student's habits of skipping classes, not preparing for quizzes and exams, and missing assignment deadlines. Such habits are not easily changed, and most students who fail the course the first time hardly pass the second time, if they passed at all.

Figure 21 shows that in the PP sections, 35% of the students were students who are have registered for the course during previous semesters. This likely had an

effect of increasing the failure rate, and the lower averages that the PP sections had.

Before the experiment started, it was crucial to know whether the students were willing to work in pairs. Luckily, most students did not have an issue with working with a partner, and the larger percentage of them (around 61% of all students) preferred working with a partner to working individually.

The students' preference to selecting their partners or having a partner assigned by the instructor or TA was measured to direct the selection of the pair formation method. The majority of students (90% of the students in both sections, as shown in Figure 23) indicated that they preferred to select their own partner.

The partner gender, educational level, and specialization were more relevant to the pair formation during the second semester rather than the first semester. When distributing the students during the second semester, these preferences were all taken into consideration, in order to match each student with a partner that was compatible with their indicated preferences. This was a main factor to consider while conducting this experiment, because, due to cultural restrictions, it was not the norm to have male and female students work closely together.

Figure 24 shows that 17% of the students in the PP section preferred to work with a partner of the same gender, 5% preferred to work with a partner of a different gender, and 78% did not have any preference. This allowed for a relatively flexible distribution of the students, in accordance with the other parameters.

However, during the first semester, these statistics were used to observe how the students indicated preference in the questionnaire compared to their actual partner selection.

Regarding the students' preference for their partners' programming level, most students (55% of all students) indicated that they preferred to work with a partner at the same level of programming as theirs. To determine the students' programming level, they were asked in the questionnaire to indicate the interval, which contained their grade in the Introduction to Programming course, and the number of times they registered for the Introduction to Programming course. These two questions, combined with the students' answer to whether they were registering for the Comp231 course, were used to approximate the students' programming level.

The last parameter that was considered when distributing the students in pairs during the second semester was the partner's specialization preference, which is shown in Figure 25. 44% of the students were indifferent to their partners' specialization, and did not mind working with a partner from the same specialization, or from a different one. On the other hand, 50% of the students preferred to work with a partner from the same specialization, and only 6% preferred to work with a partner from a different specialization.

From observations, and experience, the parameters regarding the partner, whether they were the gender, programming level, or specialization, did not have a big effect on the students' performance. However, it was noted in one pair, where a

male students and female students were paired together, that the male students, who has registered for the course during previous semesters, exhibited more commitment to the course, and put more effort during the lab, than he did in previous semesters. This could be the result of the student's act of proving his abilities in front of his female partner.

4.2.2 Pair Formation

Since the students in the first semester selected their own partner, they tended to select a friend to work with, and they got along well together. However, some conflicts issued between a few pairs, and one student explicitly declared that if they were to work in pairs in other courses they would rather not work with the same partner.

When comparing the students' preferences in partners that they indicated in the initial questionnaire to the characteristics of the partner that the students selected, the selection was very random. There was no pattern or similarity between the indicated preference and the selected partner. This is attributed to the students selecting their friends to be their partners.

In the second semester, pairs were selected by the instructors randomly. The change in pair formation method did not seem to have an effect on the pair dynamic within the lab session. The students in the pairs got along together, and were working together to solve the exercises. However, the instructor and TAs were approached with a request to switch partners by a couple of students.

From trying out two different methods of pair formation, and considering the results that were obtained, both approaches to pair formation achieved similar results. For this reason, we recommend following the approach which lets students select their own partners. Through this approach, students are already comfortable with their partner, and it reduces the overhead that pairing the students requires, which is a concern that was expressed by Gupta et. al. in [15].

4.2.3 In-class Performance

The video analysis on the recordings showed a number of interesting observations. The analysis studied the percentages of time spent typing, talking, smiling, and gazing off during the sessions. In addition, it showed a count of the number of times that students pointed to the screen, or asked for help from the instructor or the TA.

Figure 29 shows that in PP sections; 78% of the time was spent talking. This was because the students in the pair continuously communicated with each other. They discussed the exercise problem before starting programming, they discussed problems that encountered them while programming, and discussed possible better solutions, as shown in other previous researches[4, 44].

However, only 8% of the time in the traditional section was spent talking. Students tended to have their heads down, and their sight mostly on their screen or keyboard, and did not talk much with the other students sitting beside them.

On the other hand, students in the traditional sections tended to spend more time typing than their peers in the PP sections, with percentages of 40% in the PP section vs. 57% in the traditional section. The reason for such a difference is that students in PP sections tend to talk about the exercise before starting to program, therefore require less time typing. However, students in the traditional section spent a lot of time typing, and then erasing and modifying their solution.

Moreover, it was noted that almost during the entire time that PP students spent typing, they were simultaneously talking together. This meant that the code that was being written was continuously being revised, and errors and bugs were immediately caught and fixed, which agrees with researches done by Salleh et. al. [3], and He and Chen [8].

Students in the PP sections were smiling more often than those in the traditional sections. It was recorded that the students in the PP sections were smiling 16% of the time. On the other hand, students in the traditional section smiled only 2% of the time. This was used as an indication that students in the PP sections enjoyed their time in the lab more than students in the traditional sections. This was noted as well in previous research[3, 8-10, 17].

This significant difference helps in rejecting the null hypothesis H_{0_4}, and allows for the acceptance of an alternative hypothesis stating that PP has an effect on the students' enjoyment of the course.

Finally, Figure 29 shows that 19% of the time in the traditional sections was spent with students gazing off the screen. This could be due to their boredom, their being not concentrating, or their being stuck with an error that they did not know how to solve. On the other hand, gaze off was never noted in the PP sections in both semesters.

Students were encouraged in all sections to try to reach a solution on their own, and not to ask for help from the instructor or the TA unless they were stuck. Figure 30 shows that students in the traditional sections in both semesters asked for help twice as much as the students in the PP sections, with an average of twice within a ten-minute interval for the traditional sections, and once for the PP sections, although this difference was not significant. This can be easily attributed to the fact that when two students work together, and one of them does not know the answer to a problem, their partner is likely to have the solution. This allows them to learn from each other's experiences, as shown by Porter et. al. in [17].

The last parameter that was measured in the video analysis was the number of time the students pointed to the screen. This was never observed in the traditional sections, but was often observed in the PP section with 5.74 times on average per ten-minute interval. It is claimed that a screen with finger smudges is an indication of a successful pair-programming session[18].

4.2.4 Code Quality

Measuring the code quality is key to this experiment, because it is not enough to measure students' grades, and their perception of PP. Nine projects were submitted by students in each of the PP and traditional sections during the second semester. These projects were studied, reviewed, and analyzed to detect whether PP had an effect on students' code quality.

Using SourceMonitor, a software designed to produce certain statistics about code in a number of programming languages, the following observations were made:

- PP students wrote shorter code, with their code length averaging at 109 lines, and the traditional section's code averaging at 296 lines. Similarly, the number of statements in the PP code was 132 lines on average, and the traditional code was 191 lines, as indicated in Figure 31. Keating explains in [55] that shorter and simpler code is generally a better code.
- PP students wrote code with more classes (7.63 classes per program on average), when compared to the code written by students in the traditional section (6.89 classes per program on average), as shown in Figure 32. This indicates that PP students were able to modularize their code better than the traditional section students. In addition, using methods promotes code reusability, and simplicity, as indicated by Swartz in [56].
- The number of methods in the traditional section students' code was larger than the number of methods in the PP section, as per Figure 32. However,

when reviewing the code, it was noticed that this increase in average was due to one student having empty bodied methods that resulted from auto generating mouse action listeners.

- On average, PP section students' code contained less than one error (syntax or compilation) per program, while the traditional section students' code contained on average almost five syntax or compilation errors, as illustrated in Figure 33.
- PP section students commented their code more than regular section students. SourceMonitor indicated that PP section code was 10% comments, while the traditional section code was only 6%, as seen in Figure 34. However, a large percentage of the comments in the traditional section code were auto generated TODO comments, which were produced by from auto generating the mouse action listener, as mentioned above.
- The depth of the code written by students in the PP section averaged on 3, while the depth of the code written by students in the traditional section averaged on 3.45, which is shown in Figure 35.
- In correlation with the code depth, the code produced by PP students was evaluated to be less complex than the code produced by students in the traditional section, as shown in Figure 35. A complex code is harder to modify or debug, as shown in [55].
- The overall code correctness was measure by the average grades that the students received for their code, which is plotted in Figure 36. The PP

section students obtained on average 77.5% on their programs, and the traditional section students got 75.6%.

These results help us reject the null hypothesis H_{0_1}, allowing us to accept an alternative hypothesis that states that the use of PP has a positive effect on the quality of the code that the students produce.

4.2.5 Course Assessment

The course assessment depended on the grades of four quizzes, four assignments, a midterm, a practical exam, and a final. These grades were distributed the same throughout both semesters.

Regarding quizzes, the averages varied throughout the semesters, with the PP sections getting higher averages at times, and the traditional sections getting better averages at other times. However, the results of the t-test that was administered on this data, and illustrated in Figure 35, show that the differences between the two sections' averages were significant in the first and third quizzes, as well as in the quizzes total.

The reason that the differences were not always significant is the number of students who missed the quizzes due to skipping lab sessions in the traditional section. This meant that mostly serious and hard working students were present for the quiz, resulting in a better average than what would have been had all the students been present.

Likewise, the assignment results, as illustrated in Figure 36, were not substantially different between the PP and traditional sections, with the exception of the third assignment. This is understandable, since students worked on assignments at home, taking their time, and using whatever resources they required. In addition to that, the number of students that submitted their assignments was more in the PP section, while students in the traditional sections preferred to forego the submission of assignments they had trouble solving, rather than trying to find a solution or submitted work that was not complete. This might be an indication of the students' persistence and confidence in their ability to solve problems, which PP is believed to enhance.

However, the assignments total showed a significant difference between the PP and traditional sections. This could be due to the fact that several students in the traditional section did not submit one or more assignment, as mentioned previously, resulting in a low assignments total, even though individual assignments had good grades.

The practical exam results, shown in Figure 37, were fairly close in both semesters, and the t-test showed that they did not have much significant. This is due in the most part to the withdrawal of students from the traditional section. The weakest and most unconfident students were noticed to have withdrawn from the course during the period between the midterm exam and the practical exam. Therefore, the number of weak students in the PP sections was more than those in

the traditional section, which affected the averages, and the significance of the difference in the results.

The lab total were significantly better in the PP section, as seen in Figure 38, which can be reasoned by the fact that the students in the PP section had more confidence in their work and were getting better grades than the students in the traditional section. In addition, they did not miss out on quizzes and assignments throughout the semester, resulting in better totals than those in the traditional sections.

It can be noted that according to Figure 39, the results of the PP students in the midterm were significantly better than those of the traditional section students. However, since most withdrawals take place between the midterm and the practical exam time, the results of the final exam were not significantly different. Nevertheless, in the overall total of the semester, the difference between the PP section's results, and the traditional section's results is significant with an 85% confidence.

The significant difference in the students' grades allows us to reject the null hypothesis H_{0_2}, and accept the alternative hypothesis that states that PP has the ability to improve the grades that the students get in the course.

Due to their higher confidence in their abilities, the number of students who withdrew from the course in the PP sections was almost half of the number of students who withdrew from the course in the traditional sections. This can be

attributed to the increase in the students' confidence in their ability to complete the requirements of the course, and their will to commit to it. This result comes in accordance with the findings of [3, 8-10, 17]. Likewise, the low absence rate can be caused by the enjoyment the students had in the PP labs and classes.

However, since less students drop out of the course, the chances of having students who are weaker increase, making it more likely to have some students, although still a low percentage, fail the course. This resulted in having the fail rate a little bit higher in the PP section than in the traditional section.

These percentages allows us to reject the null hypothesis H_{0_3}, and accept the alternative hypothesis that states that PP improves the students course completion rate.

4.2.6 Post-Experiment Questionnaire

The post-experiment questionnaire was designed to measure what students thought of PP and of the experiment, they underwent during the semester. It asked questions about the effect of PP on the course, in terms of their understanding concepts better, solving problems and assignments faster, improving the learning experience, and making it more fun. The answers to all these parameters were positive in both semesters.

As shown in Figure 40, 82% of all students in both PP sections felt that working with a partner in PP setting helped them understand concepts that were not clearly

understood during lecture time. This is understandable, since partners helped each other while they were doing the exercises in the lab, including explaining any concepts or ideas that are not understood. Moreover, students to learn better from their peers than they do from instructors as illustrated in [17].

Since two minds are better than one, working with a partner was thought to help students solve the lab exercises faster, according to 73% of the students. Every students on their own will have to spend some time thinking, looking back through their note, and programming, then re-programming. When working with a partner, whenever stuck, the partner will likely have a solution, or an idea that will have the workflow go along. In addition to that, since with a partner the code was continuously being revised while it was written, making mistakes that will lead to rewriting an entire program became less likely in PP sections. This was observed by the researcher as well.

This reflected as well on solving individual assignments, even if only with the agreement of 55% of the students in the PP sections. This can be attributed to the fact that when working with a partner, students gain experience and knowledge that they would not have been exposed otherwise, increasing their ability to solve exercises and assignment individually, even if not to the extent of solving lab exercises together with a partner.

When asked whether they felt that PP made programming more enjoyable, 88% of the PP students answered affirmatively. This was reflected as well in the video

analysis, where PP students were shown to smile more than their peers in the traditional sections.

Finally, students were asked if they thought that PP was able to improve their learning experience generally, and 82% of the students felt that it did.

The second part of the questionnaire was targeted to measure the compatibility between partners, and how they perceived working with them. The results of these questions are shown in Figure 41. The questions were targeted to measure whether the students learnt new concepts from their partners, got a long, and distributed the work equally.

The question of whether the students learnt new concepts for their partners relates with the questions of whether PP helped students understand ambiguous concepts. However, this question is focused more on the partners helping each other, and 71% of the students felt that they were able to learn something new from their partner, regardless of whether it was a concept that was explained in the lecture, or an entirely new idea.

When asked about the understanding between partners, 82% of the students felt that they got along well, and did not have any conflicts. However, as mentioned above, there were some conflicts between students, and the instructors were approached with requests to switch partners in several cases, though they were a minority.

The following questions asked whether the students switched roles regularly, and whether they felt that the work was distributed equally among the partners. 71% of the students felt that the switched roles regularly. However, 17% felt that they did not. This was because some students explicitly declared that they preferred to be in the navigator role, and would rather not be the ones typing on the keyboard.

Similarly, 69% of the students felt that the work was divided equally among them and their partners, while 19% felt that the work was not divided equally, whether it was because their partner relayed on them to solve the exercises more, or whether they had a controlling partner that would not relinquish the keyboard and take the navigator role.

Finally, when asked about whether they felt that their partner was of the same academic level as they were, only 37% agreed with that statement, while 39% disagreed. This was expected, because even though most students indicated that they would prefer to work with a partner at the same academic level as they are, abilities vary, and students develop at different paces.

When asked whether the students felt that they benefited more when assuming the role of the driver or the navigator, 49% of the students felt that they benefited as drivers, and only 18% as navigators. This is believed to stem from every individual's learning style, whether they prefer learning while doing, making them prefer the driver role, or learning by example, making them lean towards being navigators.

Finally, a key question was whether the students would like to work with PP in other semesters, and 82% of the students answered in agreement. Regardless of all the statistics collected from this questionnaire, this question has a significant weight. Measuring according to this question indicates that the experiment was successful, since it made the students want to try it again in other courses.

This allows to reject the null hypothesis H_{0_4}, and accept the alternative hypothesis pointed in section 4.2.3, stating that PP has an effect on increasing the students' enjoyment of the course.

4.2.7 Observations

General impressions are more difficult to measure, but informal chats with the students during the semester and the instructors' observations indicated that:

- Students in the PP section were enjoying the lab more.
- Students in the PP section had more confidence in the code they were producing, whether it was for in-class tasks, quizzes, or assignments.
- Students in the PP section often came to the lab with prior knowledge of the tasks that they were going to take, and prepared accordingly.
- Students in the PP section were more interested in coming up with different ideas for programs to write.
- Some students in the PP section programmed simple games, and converted them to android applications.

These observations were all based on the researcher's past experience as a TA for various CS courses.

Chapter 5 Conclusion and Future Work

The work described in this thesis was concerned with the implementation of pair-programming as a teaching technique, which has a potential of improving the learning experience of programming languages students in the Palestinian universities. From the experience that was carried out in BZU during the academic year 2014 – 2015, this research comes to the conclusion that PP proved to have many merits that were previously discussed in the literature.

5.1 Conclusion

This research concludes that PP has a potential to improve the overall performance of the students in advanced programming courses, allowing us to reject the null hypothesis H_{0_1}. One of the aspects that can be improved by PP is the quality of the code that is produced by the students, in accordance with the research done in [7, 15, 16]. Students in the pair-programming sections usually produced a code that had fewer errors, and was simpler and of better quality.

Another aspect that may be improved through PP is the students' performance in the Comp231 course, in terms of grade, in accordance with the research done by Porter in [17]. This allowed us to reject the null hypothesis H_{0_2}. Students in the PP sections scored significantly better than their peers in the traditional sections, in a number of exams and assignments, and more importantly in the overall averages of the course.

Moreover, PP has the potential to increase the students' completion rate in the Comp231 course, and reduced the absence rate in the sections that were taught using the PP technique, in par with [8, 14, 15, 18], leading to the rejection of the null hypothesis H_{0_3}. Students who are enjoying the course, and have more confidence in their programming abilities tend to avoid dropping the course, and try to complete all the requirements needed to pass the course.

Finally, PP has the ability to increase the enjoyment students in the Comp231 course, as hypothesized by [14, 15]. This allowed for the rejection of the null hypothesis H_{0_4}. PP forces students to interact with each other, allowing them to socialize within the classroom. This leads to students feeling more relaxed, allowing them to enjoy the lab sessions more.

Moreover, some benefits of PP, that were noted during the research's presentation in conferences, is lowering the number of devices needed in a lab room. When every pair works on one device, the existing devices could be distributed to serve twice as many students as they currently do.

In addition, it was noted that PP may have the potential to pave the path for students to transfer from university life to the industry more smoothly. This is due to the fact that they would already be comfortable working in groups, and are accustomed to distributing the workload among a number of people.

5.2 Future Work

Although the results presented in this thesis demonstrate that PP has a potential of succeeding as a teaching technique, further research and experiment might be required.

A future study may be needed to measure the effect of PP on the students' ability to work in teams in the future. Even though this study indicates that the students were able to perform in pairs for the duration of the semester, a longer study is needed to allow for measuring the participants' ability to work in teams, and the role that PP plays in that.

Studying the possibility of applying PP as a teaching technique in other programming courses such as the Introduction to Programming course in BZU, as well as the possibility of applying PP as a teaching technique in other universities than BZU, may be studied in the future. In addition, the potential of applying PP techniques in courses that do not focus mainly on programming may be explored.

Finally, in order to bridge the gap between education and the industry, it may be of merit to study the potential of applying PP techniques in the Palestinian software development industry. This may be of interest since there are some companies in the Palestinian industry that indicated, as shown in Table 1, that they do apply PP principles in their work.

Bibliography

1. Begel, A. and N. Nagappan. *Pair programming: what's in it for me?* in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. 2008. ACM.

2. Wikipedia. *Pair Programming*. 2014 [cited 2014 November 26]; Available from: https://en.wikipedia.org/wiki/Pair_programming.

3. Salleh, N., E. Mendes, and J. Grundy, *Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review*. Software Engineering, IEEE Transactions on, 2011. **37**(4): p. 509-525.

4. Teague, M.M., *Pedagogy of introductory computer programming: a people-first approach.* 2011.

5. Gómez, O.S., J.L. Batún, and R.A. Aguilar, *Pair versus Solo Programming--An Experience Report from a Course on Design of Experiments in Software Engineering.* arXiv preprint arXiv:1306.4245, 2013.

6. Khan, S., et al. *A Pair Programming Trial in the CS1 Lab*. in *Proc. Annual International Conference on Computer Science Education: Innovation and Technology (CSEIT)*. 2010.

7. Prabhakar, A.B. *Applying pair programming for advanced Java course: a different approach*. in *Proceedings of the 2011 conference on Information technology education*. 2011. ACM.

8. He, X. and Y. Chen, *Analyzing the Efficiency of Pair Programming in Education.* 2015.

9. McDowell, C., et al. *The effects of pair-programming on performance in an introductory programming course*. in *ACM SIGCSE Bulletin*. 2002. ACM.

10. Salleh, N., et al. *The effects of neuroticism on pair programming: an empirical study in the higher education context*. in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. 2010. ACM.

11. Mendes, E., L. Al-Fakhri, and A. Luxton-Reilly. *A replicated experiment of pair-programming in a 2nd-year software development and design computer science course*. in *ACM SIGCSE Bulletin*. 2006. ACM.

12. Bennedsen, J. and M.E. Caspersen, *Failure rates in introductory programming.* ACM SIGCSE Bulletin, 2007. **39**(2): p. 32-36.

13. Depradine, C.A., *Using gaming to improve advanced programming skills.* The Caribbean Teaching Scholar, 2012. **1**(2).

14. Wood, K., et al., *It's Never Too Early: Pair Programming in CS1.*

15. Gupta, S., V. Bhattacharya, and M. Singha, *Pair Programming "Potential Benefits and Threats".* International Journal of Advanced Computer Research, 2013. **3**(1).

16. Radermacher, A., G. Walia, and R. Rummelt. *Assigning student programming pairs based on their mental model consistency: an initial investigation*. in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. 2012. ACM.

17. Porter, L., et al., *Success in introductory programming: what works?* Communications of the ACM, 2013. **56**(8): p. 34-36.

18. Wray, S., *How pair programming really works.* IEEE software, 2010. **27**(1): p. 50-55.

19. Taji, D. and M. Nawahdah, *Enhancing Computer Learning Activities Using Pair-Programming Techniques.*

20. Cockburn, A. and L. Williams, *The costs and benefits of pair programming.* Extreme programming examined, 2000: p. 223-247.

21. Beck, K., *Embracing change with extreme programming.* Computer, 1999. **32**(10): p. 70-77.

22. Bruegge, B. and A.H. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns and Java-(Required)*. 2004: Prentice Hall.

23. Müller, T., et al., *Certified Tester Foundation Level Syllabus.* Journal of International Software Testing Qualifications Board, 2011.

24. Schmalz, M.S. *Engineering Design Process*. Design 1 2015 [cited 2015 July 4]; Available from: www.cise.ufl.edu/~mssz/Design1/CEN3913-Lecture01-Design-ProcessDesign.html.

25. DewsoftOverseas. *Software Development*. Dewsoft Software Engineering Tutorial 2000 [cited 2015 July 4]; Available from: education.dwesoftoverseas.com/QE/QuickReference/Software%20Engineering/2.3.asp.

26. Sparrow, P. *Spiral Model: Advantages and Disadvantages*. 2011 [cited 2015 July 4]; Available from: www.ianswer4u.com/2011/12/spiral-model-advantages-and.html#axzz3eupBCCAm.

27. Sillitti, A., G. Succi, and J. Vlasenko. *Understanding the impact of Pair Programming on Developers Attention*. in *Proceedings of the ICSE2012 Conference, Zurich, Switzerland*. 2012.

28. Highsmith, J., *Agile software development ecosystems*. 2002: Addison-Wesley Longman Publishing Co., Inc.

29. Beck, K., et al., *Manifesto for agile software development.* 2001.

30. Beck, K., *Extreme programming explained: embrace change*. 2000: Addison-Wesley Professional.

31. Bain, L. *Try Pair Programming*. 2015 [cited 2015 1 May]; Available from: https://developer.atlassian.com/blog/2015/05/try-pair-programming/.

32. Lui, K.M., K.A. Barnes, and K.C. Chan, *Pair Programming: Issues and Challenges*, in *Agile Software Development*. 2010, Springer. p. 143-163.

33. Balijepally, V., et al., *Are two heads better than one for software development? The productivity paradox of pair programming.* Management Information Systems Quarterly, 2009. **33**(1): p. 7.

34. di Bella, E., et al., *Pair Programming and Software Defects--A Large, Industrial Case Study.* Software Engineering, IEEE Transactions on, 2013. **39**(7): p. 930-953.

35. Giri, M. and S. Soni, *Effectiveness of Software Development Process Using Programmer Ranker Algorithm in Pair Programming.* 2013.

36. Ambler, S.W., *Survey says... agile has crossed the chasm.* DR DOBBS JOURNAL, 2007. **32**(8): p. 59-61.

37. Conradi, R. and M.A. Babar, *Controlled Experiments on Pair Programming: Making Sense of Heterogeneous Results.* Norsk informatikkonferanse (NIK), 2010.

38. Patel, D., U. Panchal, and K. Nagpal, *When and Where of Pair Programming: A literature review.*

39. Corbett, A., M. Holcombe, and S.J. Wood, *It's the People, Stupid!-Formal Models of Social Interaction in Agile Software Development Teams.* 2015.

40. Akinola, O.S., *An Empirical Comparative Analysis of Programming Effort, Bugs Incurrence and Code Quality Between Solo and Pair Programmers.* Middle-East Journal of Scientific Research, 2014. **21**(12): p. 2231-2237.

41. Salinger, S., F. Zieris, and L. Prechelt. *Liberating pair programming research from the oppressive driver/observer regime*. in *Proceedings of the 2013 International Conference on Software Engineering*. 2013. IEEE Press.

42. Inoue, T., *Investigating the Relation between Behavior and Result in Pair Programming: Talk and Work Leads to Success.* The Journal of Information and Systems in Education, 2013. **12**(1): p. 39-49.

43. Simon, B. and B. Hanks, *First-year students' impressions of pair programming in CS1*. Journal on Educational Resources in Computing (JERIC), 2008. **7**(4): p. 5.

44. Bailey, R. and E. Mentz, *IT Teachers' Experience of Teaching–Learning Strategies to Promote Critical Thinking.* Issues in Informing Science and Information Technology, 2015. **12**.

45. Explorable, *Experimental Research*. 2008 - 2015: Explorable.com.

46. MacKenzie, I.S., *Human-computer interaction: An empirical research perspective*. 2012: Newnes.

47. Salleh, N., E. Mendes, and J. Grundy, *Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments.* Empirical Software Engineering, 2014. **19**(3): p. 714-752.

48. Sloetjes, H. and P. Wittenburg. *Annotation by Category: ELAN and ISO DCR*. in *LREC*. 2008.

49. Brugman, H., A. Russel, and X. Nijmegen. *Annotating Multi-media/Multi-modal Resources with ELAN*. in *LREC*. 2004.

50. Alemerien, K. and K. Magel. *Experimental Evaluation of Static Source Code Analysis tools*. in *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*. 2013. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

51. Organisation, I.S., *Information Technology-Software Product Evaluation: Quality Characteristics and Guidelines for their Use*. ISO/IEC IS 9126, Geneva, 1991.

52. Wyllys, R.E., *THE UNIVERSITY OF TEXAS AT AUSTIN SCHOOL OF INFORMATION*.

53. Wikipedia. *Student's T-Test*. 2014 [cited 2014 December 28]; Available from: https://en.wikipedia.org/wiki/Student's_t-test.

54. Software, C., *SourceMonitro*. 2011.

55. Keating, M. *Good Design of Impossibly Complex Chips*. 2015.

56. Swartz, F. *Commentary: Why use methods?* 2006 [cited 2015 10 July]; Available from: http://www.fredosaurus.com/JavaBasics/methods/method-commentary/methcom-purpose.html.

57. Inc., T.M. *t Table*. Normal Table 2011 [cited 2014 29 December]; Available from: http://www.normaltable.com/ttable.html.

Appendix I – List of Emailed ICT Institutes

1. Al-Haitham for Technology Development (HTD)
2. eGate for Self- service Solutions
3. Good Shepherd Engineering
4. PCNC 2000 Networking Co.
5. Transcend Support
6. CloudTech Solutions
7. AXSOS AG
8. 2i Software
9. 2M Solutions Security Group
10. Abbmatrix
11. Akram Sbitany and Sons Co. Ltd.
12. Al Jarmaq For computer & Electronic Services
13. Al Nasher Technical Services
14. Al-andalus Software Development (ASD)
15. Al-Canaan Group
16. Al-Jaffal Group for Trade & Investment
17. Art Technologies
18. ASAL Technologies
19. Axizo
20. BabilSoft Information Systems
21. Badawi Information Systems
22. BCI
23. Bisan Systems Ltd.
24. Business Intelligence Technologies Co.
25. Call U ISP
26. Computer & Communications Systems (CCS)
27. Computer Media Center C.M.C
28. Coolnet Internet Service Provider
29. CPS - Creative Programming Solutions

30. Cystack
31. DataSet Software Solutions
32. Dimensions
33. Dimensions Studio
34. Electronic Digital Information Systems LTD
35. ERICSSON RADIO SYSTEMS AKTIEBOLAGET
36. Exalt Technologies
37. Experts Turnkey Solutions
38. Galaxy Information Systems
39. Ghatasheh Technology & Investment Ltd.
40. GlobalCom Telecommunications Plc.
41. GPAL
42. Hadara Technologies
43. Harmony Solutions
44. Headway Academy for Training & Development
45. Hulul for Business Solutions
46. iConnect Tech
47. ID Management Consultants
48. Infinity Information Technology
49. Intertech Co. for Computer and Internet services
50. Intracom
51. Iris Interactive Solutions
52. Jaffa.Net Software and Communications
53. Jawwal
54. Jerusalem Technology
55. Jordan Business Systems (JBS)
56. Mada Al Arab
57. Madar Telecommunication Copany
58. MaDeK
59. Massar Associates
60. MobiStine

61. Modern Arabian Business Corporation- MABCO
62. National Computers & Software Co. Ltd.
63. National Computing Resources (NCR)
64. Neiraba Animation Studios
65. Newsoft For Programming & Information Technology
66. NTS
67. office world
68. Pal Power Electronic Engineering & Energy
69. Palestine Office Technology - OFFTEC
70. PalPay
71. Paltech
72. Primus - Computer Networking Services
73. ProGineer Technologies
74. Reach
75. SABRI FOR COMPUTER
76. Safad Engineering & Electronics
77. Sector Security Group Technology
78. SocialDice
79. Software Technology Development & Data Processing (STDDP)
80. Specialized Technical Services - STS
81. SteadyPoint
82. Technology plus
83. Technopal for Engineering & Communication
84. ULTIMIT Advanced Turnkey Solutions
85. Wataniya Mobile
86. Zaytona Telecomunication
87. zone technologies
88. شركة عالم المستقبل التجارية
89. Informatica
90. Tamkeen for Information Technology
91. Trusted Systems for Computer and IT

92. Dot Media
93. DAMAN for International Trading Sevices
94. Global Tec For Training & Computes
95. Infinite Tiers Group, Inc.
96. Inter. Telecom. & Elect. Corporation "ITEC"
97. Isra Software & Computer Co.
98. millennium technology
99. MMS Software Solutions
100. Paltel - Palestine Telecommunications Company
101. PITS
102. Spark Consulting & Training
103. شركة كومباكت للكمبيوتر والالكترونيات

Appendix II – Email sample – the ICT Sector

from: Dima Taji <dtaji@birzeit.edu>

to: omar.kamal@progineer.net

cc: "Ma'moun I. Nawahdah" <mnawahdah@birzeit.edu>

subject: Pair-programming Research Question

Dear Mr Omar Kamal,

My name is Dima Taji. I am a teaching assistant in the Computer Science Department at Birzeit University.

I am currently pursuing my Master Degree in Computing. My master thesis is about “Pair-programming, and its adaptability in the Palestinian industry and education sectors”.

At your earliest convenience, could you please answer the following question:
Does any team in your respected company apply “Pair-programming” technique in their projects?

Looking forward to hearing from you,

-- Dima

Appendix III Initial Questionnaire

الرقم الجامعي: _____

الجنس: ☐ **ذكر** ☐ **أنثى**

- من أي منطقة أتيت:
  - ☐ شمال
  - ☐ وسط
  - ☐ جنوب

- المدرسة التي أنهيت منها صف التوجيهي كانت:
  - ☐ مدرسة حكومية
  - ☐ مدرسة خاصة مختلطة
  - ☐ مدرسة خاصة غير مختلطة

- كم كان معدلك التوجيهي:
  - ☐ أقل من 70
  - ☐ 75 – 70
  - ☐ 80 – 76
  - ☐ 85 - 81
  - ☐ 90 – 86
  - ☐ 95 – 91
  - ☐ 99 – 96

- التخصص:
  - ☐ علم حاسوب
  - ☐ هندسة أنظمة حاسوب
  - ☐ غير ذلك: _____

- خلفية برمجة لغة الC:
  - ☐ 132
  - ☐ 142
  - ☐ 230
  - ☐ غير ذلك: _____

- كم مرة أخذت مساق برمجة لغة الC في السابق:
  - ☐ مرة
  - ☐ مرتين
  - ☐ أكثر من مرتين

- كم كان معدلك في أخر مساق C أخذته:
  - ☐ أقل من 68
  - ☐ 68 – 72
  - ☐ 73 – 77
  - ☐ 78 – 84
  - ☐ 85 – 89
  - ☐ 90 – 99

- هل هذه أول مرة تسجل فيها لمساق البرمجة المتقدمة: ☐ نعم ☐ لا
- هل تفضل العمل المنفرد أم ضمن مجموعة: ☐ منفرد ☐ ضمن مجموعة ☐ لا يوجد فرق
- أفضل اختيار شريكي بالعمل بنفسي: ☐ أوافق ☐ لا أوافق

- أي من الجمل التالية تنطبق عليك (رجاء اختر جواب واحد من كل مجموعة):

○ أفضل العمل مع شخص من نفس مستواي الأكاديمي	○ أفضل العمل مع شخص من مستوى أكاديمي اقل
○ أفضل العمل مع شخص من مستوى أكاديمي أعلى	○ لا يشكل المستوى الأكاديمي فرقا بالنسبة لي
○ أفضل العمل مع شخص من الجنس الأخر	○ أفضل العمل مع شخص من نفس الجنس
○ لا يشكل الجنس فرقا بالنسبة لي	
○ أفضل العمل مع شخص من تخصص مختلف	○ أفضل العمل مع شخص من نفس التخصص
○ لا يشكل التخصص فرقا بالنسبة لي	

Appendix IV Follow-up Questionnaire

من خلال تجربتك بالبرمجة الثنائية خلال مساق البرمجة المتقدمة 231، الرجاء الإجابة على الأسئلة التالية:

1. ساعدني العمل مع شريك في فهم بعض المفاهيم التي لم تكن واضحة خلال المحاضرات:
   1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

2. تعلمت أشياء ومفاهيم جديدة من شريكي بالعمل
   1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

3. ساعدني العمل مع شريك على إتمام حل الأسئلة بسرعة أكبر
   1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

4. ساعدني العمل مع شريك على القيام بالواجبات والمهام الفردية بشكل أحسن
   1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

5. جعل العمل مع شريك عملية البرمجة أكثر متعة
   1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

6. كان هناك تفاهم بيني وبين شريكي بالعمل
   1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

7. كنا نتبادل أنا وشريكي الأدوار بين المبرمج (driver) والموجه (navigator) بشكل منتظم
   1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

8. كان هناك تقسيم متساو للعمل بيني وبين شريكي
   1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

9. أحسست بأنني كنت أستفيد بشكل أكبر عندما كنت
   1. المبرمج (driver) 2. الموجه (navigator) 3. لم يكن هناك فرق

10. أنا وشريكي من نفس المستوى الأكاديمي برمجيا
    1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

11. تجربتي بالتعلم من خلال العمل مع شريك كانت أفضل
    1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

12. أود أن أتعلم من خلال العمل مع شريك في مساقات أخرى
    1. أوافق بشدة 2. أوافق 3. حيادي 4. لا أوافق 5. لا أوافق أبدا

Appendix V T-Test Table[57]

df	$t_{.80}$	$t_{.90}$	$t_{.95}$	$t_{.975}$	$t_{.99}$	$t_{.995}$
1	1.376	3.078	6.314	12.71	31.82	63.66
2	1.061	1.886	2.920	4.303	6.965	9.925
3	.978	1.638	2.353	3.182	4.541	5.841
4	.941	1.533	2.132	2.776	3.747	4.604
5	.920	1.476	2.015	2.571	3.365	4.032
6	.906	1.440	1.943	2.447	3.143	3.707
7	.896	1.415	1.895	2.365	2.998	3.499
8	.889	1.397	1.860	2.306	2.896	3.355
9	.883	1.383	1.833	2.262	2.821	3.250
10	.879	1.372	1.812	2.228	2.764	3.169
11	.876	1.363	1.796	2.201	2.718	3.106
12	.873	1.356	1.782	2.179	2.681	3.055
13	.870	1.350	1.771	2.160	2.650	3.012
14	.868	1.345	1.761	2.145	2.624	2.977
15	.866	1.341	1.753	2.131	2.602	2.947
16	.865	1.337	1.746	2.120	2.583	2.921
17	.863	1.333	1.740	2.110	2.567	2.898
18	.862	1.330	1.734	2.101	2.552	2.878
19	.861	1.328	1.729	2.093	2.539	2.861
20	.860	1.325	1.725	2.086	2.528	2.845
21	.859	1.323	1.721	2.080	2.518	2.831
22	.858	1.321	1.717	2.074	2.508	2.819
23	.858	1.319	1.714	2.069	2.500	2.807
24	.857	1.318	1.711	2.064	2.492	2.797
25	.856	1.316	1.708	2.060	2.485	2.787
26	.856	1.315	1.706	2.056	2.479	2.779
27	.855	1.314	1.703	2.052	2.473	2.771
28	.855	1.313	1.701	2.048	2.467	2.763
29	.854	1.311	1.699	2.045	2.462	2.756
30	.854	1.310	1.697	2.042	2.457	2.750
40	.851	1.303	1.684	2.021	2.423	2.704
60	.848	1.296	1.671	2.000	2.390	2.660
80	.846	1.292	1.664	1.990	2.374	2.639
100	.845	1.290	1.660	1.984	2.364	2.626
120	.845	1.289	1.658	1.980	2.358	2.617
∞	.842	1.282	1.645	1.960	2.326	2.576

Appendix VI ELAN Output for the PP Section

PP Section			
Tier	Start Time	Finish Time	Duration
Typing	00:14.2	00:20.5	00:06.3
Typing	00:58.5	01:59.3	01:00.8
Typing	02:25.7	02:45.4	00:19.7
Typing	04:57.4	05:19.7	00:22.3
Typing	10:15.5	10:46.0	00:30.5
Typing	11:47.0	11:56.1	00:09.1
Talking	00:00.0	00:14.2	00:14.2
Talking	00:20.4	00:58.3	00:37.9
Talking	02:06.0	04:57.5	02:51.5
Talking	04:57.5	06:32.6	01:35.1
Talking	06:32.6	11:56.1	05:23.5
Smiling	01:25.9	01:32.5	00:06.6
Smiling	01:59.8	02:06.3	00:06.5
Smiling	03:25.4	04:37.4	01:12.0
Smiling	05:36.5	06:32.7	00:56.2
Smiling	07:56.0	08:42.4	00:46.4
Smiling	08:49.8	10:14.8	01:25.0
Smiling	10:15.5	10:46.0	00:30.5
Pointing at Screen	00:20.4	00:23.4	00:03.1
Pointing at Screen	00:25.4	00:26.7	00:01.3
Pointing at Screen	00:37.9	00:43.6	00:05.7
Pointing at Screen	00:55.2	00:58.9	00:03.7
Pointing at Screen	02:02.4	02:04.9	00:02.5
Pointing at Screen	03:13.6	03:18.4	00:04.8
Pointing at Screen	03:27.1	03:30.3	00:03.2
Pointing at Screen	10:46.6	11:05.5	00:18.9
Talking	00:01.7	00:29.0	00:27.3
Talking	01:00.8	02:34.2	01:33.4
Talking	02:57.8	04:39.7	01:41.9
Talking	04:39.8	05:36.3	00:56.5
Talking	05:44.1	06:44.2	01:00.1
Typing	00:29.2	01:00.6	00:31.4
Typing	01:13.7	01:16.5	00:02.7
Typing	01:18.6	01:28.0	00:09.4
Typing	01:33.9	01:43.6	00:09.7
Typing	01:55.1	02:04.0	00:08.9
Typing	02:33.7	02:57.3	00:23.6
Typing	04:09.4	04:40.1	00:30.7
Typing	05:33.6	05:43.5	00:09.9
Typing	06:05.4	06:44.2	00:38.8

Smiling	00:39.5	00:56.9	00:17.4
Pointing at screen	02:24.8	02:33.3	00:08.5
Pointing at screen	02:45.9	02:54.0	00:08.1
Talking	00:00.3	01:04.0	01:03.7
Talking	01:04.1	10:09.7	09:05.6
Typing	00:21.2	00:43.9	00:22.7
Typing	01:52.4	02:12.1	00:19.7
Typing	02:25.6	02:43.9	00:18.3
Typing	05:05.8	05:49.0	00:43.2
Typing	06:06.2	07:03.0	00:56.8
Typing	07:17.6	07:28.1	00:10.5
Typing	07:34.5	07:36.9	00:02.4
Typing	08:05.7	08:47.2	00:41.5
Typing	09:29.3	10:09.7	00:40.4
Smiling	00:00.3	01:04.0	01:03.7
Smiling	01:22.6	01:52.2	00:29.6
Smiling	03:36.8	04:26.2	00:49.4
Smiling	08:42.6	09:29.6	00:47.0
Talking	01:15.2	02:39.6	01:24.3
Talking	03:10.4	05:28.8	02:18.3
Talking	05:38.1	06:15.3	00:37.2
Talking	06:33.9	10:14.3	03:40.4
Typing	02:36.9	03:10.7	00:33.9
Typing	04:11.2	04:49.6	00:38.4
Typing	05:23.8	05:37.9	00:14.1
Typing	06:10.4	06:37.0	00:26.6
Typing	07:04.0	07:18.2	00:14.2
Typing	08:20.5	10:05.9	01:45.4
Smiling	01:13.1	01:19.7	00:06.6
Smiling	03:05.3	03:22.9	00:17.6
Smiling	04:04.6	04:11.1	00:06.4
Smiling	05:09.4	05:16.5	00:07.0
Smiling	07:49.1	08:09.7	00:20.6
Asking for help	00:00.0	01:15.1	01:15.1
Typing	00:00.2	00:17.6	00:17.4
Typing	00:24.1	00:45.5	00:21.4
Typing	03:08.8	03:24.1	00:15.3
Typing	06:31.7	06:36.3	00:04.6
Typing	07:12.1	07:40.4	00:28.3
Typing	07:40.8	09:49.6	02:08.8
Talking	00:17.6	00:23.9	00:06.3
Talking	00:27.2	02:10.9	01:43.7

Talking	02:11.8	03:08.8	00:57.0
Talking	03:24.4	05:15.7	01:51.3
Talking	05:17.1	05:56.2	00:39.1
Talking	06:36.7	07:11.5	00:34.8
Talking	07:40.8	09:49.6	02:08.8
Smiling	02:05.5	02:18.8	00:13.3
Smiling	05:24.5	05:43.3	00:18.8
Smiling	06:39.6	06:44.1	00:04.5
Asking for help	05:56.6	06:31.7	00:35.2
Pointing at Screen	00:49.7	00:57.6	00:07.9
Pointing at Screen	01:12.2	01:19.3	00:07.1
Pointing at Screen	01:20.8	01:26.3	00:05.5
Pointing at Screen	01:47.9	01:58.8	00:10.9
Pointing at Screen	02:36.4	02:39.1	00:02.7
Pointing at Screen	05:16.0	05:19.5	00:03.5
Talking	00:15.8	00:20.7	00:04.9
Talking	00:37.8	01:01.9	00:24.1
Talking	04:05.2	04:41.2	00:36.0
Talking	05:57.9	07:02.9	01:05.0
Talking	08:12.1	08:31.1	00:19.0
Talking	08:38.5	08:41.1	00:02.6
Talking	09:11.2	10:16.2	01:05.1
Typing	00:01.9	00:15.7	00:13.8
Typing	00:20.9	00:38.1	00:17.2
Typing	00:40.9	02:17.3	01:36.4
Typing	03:48.0	04:11.3	00:23.3
Typing	05:03.0	06:31.3	01:28.3
Typing	07:00.9	07:50.1	00:49.2
Typing	08:31.2	08:37.8	00:06.6
Typing	08:41.7	09:10.9	00:29.2
Asking for Help	02:22.0	03:34.5	01:12.5
Asking for Help	04:41.5	05:03.0	00:21.5
Asking for Help	07:49.4	08:05.9	00:16.5
Asking for Help	09:43.3	09:55.0	00:11.7
Smiling	03:35.5	03:48.1	00:12.6
Smiling	08:06.2	08:12.5	00:06.3
Pointing at Screen	07:51.0	07:53.4	00:02.4
Talking	00:02.7	00:43.1	00:40.4
Talking	00:49.3	01:24.9	00:35.6
Talking	02:05.8	02:29.1	00:23.3
Talking	02:29.4	03:43.3	01:13.9
Talking	03:43.7	04:17.7	00:34.0
Talking	04:18.1	05:46.9	01:28.8

Talking	05:47.3	07:09.2	01:21.9
Talking	07:31.0	07:35.7	00:04.7
Talking	07:36.3	07:54.9	00:18.6
Talking	08:11.8	08:42.0	00:30.2
Talking	08:44.1	10:24.3	01:40.2
Talking	10:33.3	11:03.9	00:30.6
Talking	11:15.6	11:27.7	00:12.1
Talking	12:08.4	12:08.4	00:00.1
Talking	12:08.4	13:41.8	01:33.4
Typing	00:01.1	00:02.8	00:01.7
Typing	00:42.9	00:51.5	00:08.6
Typing	01:22.9	01:23.0	00:00.1
Typing	01:23.0	02:01.9	00:38.9
Typing	02:29.4	03:43.3	01:13.9
Typing	04:18.1	05:46.9	01:28.8
Typing	07:09.1	07:31.0	00:21.9
Typing	07:54.2	08:19.5	00:25.3
Typing	08:39.5	08:43.7	00:04.2
Typing	10:24.2	10:33.0	00:08.8
Typing	11:04.2	11:15.5	00:11.3
Asking for Help	11:28.2	12:08.2	00:40.0
Pointing at Screen	02:01.9	02:05.8	00:03.9
Pointing at Screen	05:57.1	06:01.4	00:04.3
Pointing at Screen	07:31.0	07:35.7	00:04.7
Pointing at Screen	07:40.1	07:42.5	00:02.4
Pointing at Screen	08:11.4	08:14.2	00:02.8
Pointing at Screen	11:27.8	11:30.9	00:03.1
Pointing at Screen	12:43.6	12:46.9	00:03.3
Talking	00:00.0	01:28.2	01:28.2
Talking	01:46.6	01:52.7	00:06.1
Talking	02:23.7	03:31.2	01:07.5
Talking	06:01.4	06:05.4	00:04.0
Typing	01:28.3	01:32.8	00:04.6
Typing	01:32.8	01:38.2	00:05.4
Typing	01:39.0	01:46.0	00:07.0
Typing	01:53.6	02:24.1	00:30.6
Typing	02:24.2	02:32.7	00:08.5
Typing	02:34.5	02:48.2	00:13.7
Typing	02:51.9	03:03.4	00:11.5
Typing	06:05.6	06:44.7	00:39.2
Smiling	05:54.0	06:03.3	00:09.3
Pointing at Screen	00:08.0	00:10.7	00:02.7
Pointing at Screen	00:11.8	00:16.1	00:04.3
Pointing at Screen	00:32.9	00:34.6	00:01.7

Pointing at Screen	00:45.3	00:47.4	00:02.1
Pointing at Screen	00:59.9	01:02.8	00:02.9
Pointing at Screen	01:22.0	01:23.5	00:01.5
Pointing at Screen	02:32.7	02:34.6	00:01.9
Pointing at Screen	03:04.7	03:13.7	00:09.0
Pointing at Screen	03:23.3	03:26.1	00:02.9
Asking for help	03:31.2	06:00.9	02:29.6
Talking	00:00.3	04:11.6	04:11.3
Talking	04:11.8	04:48.5	00:36.7
Talking	04:48.8	05:08.8	00:20.0
Talking	05:08.9	05:27.8	00:18.9
Talking	06:05.5	06:35.7	00:30.2
Talking	06:44.9	08:45.4	02:00.5
Talking	08:45.8	10:09.6	01:23.8
Typing	00:00.3	04:11.6	04:11.3
Typing	04:48.8	05:08.8	00:20.0
Typing	05:27.9	06:05.2	00:37.3
Typing	06:36.0	06:44.7	00:08.6
Typing	08:45.8	10:09.6	01:23.8
Pointing at screen	00:12.7	00:14.6	00:01.9
Pointing at screen	00:28.5	00:30.6	00:02.1
Pointing at screen	00:54.1	00:56.8	00:02.7
Pointing at screen	02:36.1	02:42.1	00:06.0
Pointing at screen	02:57.4	02:59.5	00:02.1
Pointing at screen	03:28.6	03:31.1	00:02.4
Pointing at screen	04:42.5	04:43.5	00:01.0
Pointing at screen	08:53.1	08:55.3	00:02.2
Pointing at screen	09:12.3	09:13.5	00:01.2
Pointing at screen	09:17.1	09:17.9	00:00.8
Pointing at screen	10:07.3	10:09.6	00:02.3
Talking	00:02.0	01:44.9	01:42.9
Talking	02:12.3	02:42.5	00:30.2
Talking	04:10.1	04:26.8	00:16.8
Talking	05:02.2	05:57.8	00:55.6
Talking	05:57.8	07:07.7	01:09.9
Talking	07:08.4	07:19.3	00:10.9
Talking	07:47.0	10:20.7	02:33.7
Typing	01:12.2	01:38.0	00:25.8
Typing	01:43.2	02:14.1	00:30.8
Typing	02:38.7	04:14.0	01:35.3
Typing	04:25.1	05:04.6	00:39.5
Typing	05:57.8	07:07.7	01:09.9
Typing	07:19.4	07:47.8	00:28.4

Typing	09:53.3	10:20.7	00:27.4
Pointing at Screen	00:01.2	00:02.1	00:00.9
Pointing at Screen	00:08.7	00:10.4	00:01.7
Pointing at Screen	00:34.7	00:36.1	00:01.4
Pointing at Screen	01:38.7	01:42.5	00:03.9
Pointing at Screen	02:17.4	02:20.0	00:02.7

Appendix VII ELAN Output for the Traditional Section

Traditional Section			
Tier	Start Time	Finish Time	Duration
Typing	02:55.4	03:32.0	00:36.6
Typing	04:28.6	05:29.7	01:01.1
Typing	07:55.7	08:23.6	00:27.9
Typing	09:30.7	10:12.9	00:42.3
Talking	03:32.6	04:28.1	00:55.6
Talking	05:29.5	06:30.6	01:01.1
Talking	07:19.2	07:55.6	00:36.4
Gaze-off	02:32.6	02:54.9	00:22.3
Gaze-off	06:30.8	07:19.2	00:48.4
Asking for help	00:00.0	02:32.7	02:32.7
Asking for help	08:23.8	09:30.6	01:06.8
Talking	04:01.2	04:27.8	00:26.6
Talking	04:27.8	04:40.0	00:12.2
Talking	06:48.0	07:28.3	00:40.3
Talking	07:38.8	07:47.1	00:08.3
Typing	00:16.6	00:22.9	00:06.3
Typing	00:26.3	00:30.9	00:04.6
Typing	01:02.4	01:56.0	00:53.6
Typing	02:02.8	02:40.4	00:37.6
Typing	02:45.8	04:01.2	01:15.5
Typing	05:56.6	06:47.9	00:51.3
Typing	07:28.4	07:38.7	00:10.4
Typing	08:04.1	09:13.6	01:09.5
Smiling	00:36.7	00:42.8	00:06.1
Gaze-off	00:00.0	00:16.6	00:16.6
Gaze-off	00:23.0	00:26.3	00:03.3
Gaze-off	00:31.0	00:36.7	00:05.7
Gaze-off	00:42.5	01:02.4	00:19.9
Gaze-off	01:56.0	02:03.0	00:07.0
Gaze-off	02:40.3	02:45.9	00:05.5
Gaze-off	04:40.1	04:46.9	00:06.8
Gaze-off	07:46.9	08:04.2	00:17.3
Asking for help	04:47.0	05:56.6	01:09.6
Talking	00:00.0	00:14.5	00:14.5
Typing	02:18.7	08:40.9	06:22.2
Asking for help	00:14.4	02:18.6	02:04.2
Gaze-off	00:01.8	01:15.4	01:13.6
Gaze-off	01:28.5	01:31.3	00:02.7

Gaze-off	01:45.3	01:48.2	00:02.9
Gaze-off	03:34.3	03:40.1	00:05.8
Gaze-off	04:13.4	04:17.0	00:03.7
Gaze-off	05:07.7	05:37.6	00:29.8
Gaze-off	05:51.0	06:01.8	00:10.8
Gaze-off	06:18.1	06:27.5	00:09.4
Gaze-off	07:36.0	07:46.8	00:10.8
Typing	01:15.8	01:28.2	00:12.4
Typing	01:31.5	01:45.2	00:13.7
Typing	01:48.6	03:34.2	01:45.6
Typing	04:17.6	05:07.2	00:49.7
Typing	06:02.2	06:17.5	00:15.2
Typing	06:27.8	07:35.8	01:07.9
Typing	07:47.0	09:20.0	01:33.0
Typing	09:35.1	09:42.4	00:07.3
Talking	03:43.5	03:55.0	00:11.5
Talking	04:05.8	04:12.8	00:06.9
Talking	05:38.2	05:51.2	00:13.0
Smiling	04:07.1	04:12.7	00:05.5
Smiling	05:38.1	05:44.3	00:06.2

Appendix VIII SourceMonitor Output For the PP Section

Checkpoint Name	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods /Class	Avg Stmts /Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity
1142886	93	70	0	62	12.9	2	1	29.5	0	4	3.07	0
1132065	287	209	0.5	187	0.3	10	1	19.1	0	9+	5.96	0
1131796	262	172	2.3	112	10.7	7	3.57	4.68	5	5	2.04	1.16
1131074	208	173	12.7	144	5.8	6	1.17	19.86	17	4	1.42	5.4
1130684	236	149	0	78	16.1	19	1.32	4.04	1	4	2.08	1
1130603	216	131	0	149	12.5	10	1.1	6.73	0	9+	5.3	0
1130501	102	77	6.5	27	15.7	5	1.8	6.89	6	5	2	1.67
1121056	118	78	0	62	8.5	2	1	23.5	0	4	2.15	0

Appendix IX SourceMonitor Output For the Traditional Section

Checkpoint Name	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods /Class	Avg Stmts /Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity
1132176	905	428	12.6	416	6.9	12	2.67	12.31	7	9+	7.42	3.25
1131467	183	123	0	129	16.4	2	1	40	1	2	1.52	1
1131343	196	135	10.4	59	1	3	7	3.86	3	3	1.61	1.71
1131321	236	167	0	256	10.2	12	0.92	12.82	1	9+	5.88	1
1131273	202	130	14.6	78	5.9	1	13	6.85	14	6	2.48	2.46
1130918	145	114	7.9	58	8.3	5	2.4	6.58	6	5	1.84	2
1130606	114	81	0	68	1.8	3	0.67	35	0	5	3.55	0
1130083	569	470	18.3	250	1.1	19	1.16	20.36	99	6	3.66	10
1122211	111	72	0	73	0.9	5	0.8	14.5	0	9+	3.13	0

Appendix X Students Grades For the PP Section in the First Semester 2014 - 2015

	Quiz 1	Quiz 2	Quiz 3	Quiz 4	Quizzes Total	Assignment 1	Assignment 2	Assignment 3	Assignment 4	Assignments Total	Practical	Lab Total	Midterm	Final	Total Average
	6.00			6.00	3.00	5.60	3.67	6.00		5.38	3.00	11.38	63.00	52.00	55.00
	5.00														
	2.00	0.00				5.20	1.67						24.00		
	10.00	3.33	4.00	8.33	6.42	9.20	8.00	6.67		8.38	7.11	21.90	80.00	59.00	74.00
	5.00	1.33			1.58	8.00	5.00			4.38	1.33	7.29	58.00	33.00	55.00
	10.00	6.00	4.00	6.00	6.50			7.67	7.14	6.00	2.56	15.06	60.00	67.00	61.00
	0.00	8.67	6.00		3.67	8.00				2.50		6.17	57.00	43.00	50.00
	3.00	2.00	4.00	0.00	2.25	8.40	6.67		2.86	6.38	1.44	10.07	54.00	43.00	55.00
	4.00	1.33	9.50	6.67	5.38	4.80	9.00	8.33	8.86	11.88	7.33	24.58	77.00	70.00	77.00
	4.00	2.00	6.50	8.67	5.29	10.00		9.00	8.00	10.00	6.78	22.07	81.00	65.00	74.00
	8.00	0.00	4.00	5.00	4.25	7.20	9.17	10.00	8.57	13.19	9.78	27.22	67.00	79.00	80.00
	5.00	0.00	6.50	4.00	3.88	8.80	9.17	8.00	1.43	9.81	4.67	18.35	68.00	65.00	66.00
	5.00	6.67	6.50	8.33	6.63	8.80	9.67	10.00	10.00	14.50	9.00	30.13	69.00	81.00	82.00
	5.00	10.00	9.00	8.67	8.17	9.20	9.67	9.33	9.71	14.25	7.00	29.42	66.00	69.00	77.00
	8.00	8.00	9.00	7.33	8.08	10.00	9.67	9.00	8.00	13.63	7.44	29.15	88.00	68.00	82.00
	8.00	6.67	7.00	7.50	7.29	7.20		8.33	8.00	8.88	7.56	23.72	63.00	75.00	74.00
	10.00	8.00	8.50	0.00	6.63	8.00	9.17	10.00	9.71	13.94	6.44	27.01	69.00	67.00	75.00
	8.00	10.00	10.00	9.00	9.25	8.80	10.00	9.67	10.00	14.50	10.00	33.75	94.00	94.00	97.00
	10.00	4.67	9.00	9.67	8.33	9.60	10.00	10.00	10.00	14.88	10.00	33.21	94.00	77.00	91.00
	9.00	2.00	6.50	9.33	6.71	9.20	9.00	10.00	10.00	14.38	7.89	28.97	82.00	65.00	80.00
	10.00	8.00	9.00	8.00	8.75	8.00	9.00	9.67	10.00	13.88	9.78	32.40	90.00	86.00	93.00

	10.00	10.00	9.50	8.00	9.38	9.20	9.67	9.33		10.00	8.44	27.82	93.00	79.00	90.00
		0.67	4.50	4.00	2.29	4.80	5.00	5.00	4.57	7.25	2.00	11.54	65.00	68.00	60.00
	4.00	3.33	6.00			10.00	9.67						53.00		
	9.00	10.00	10.00	1.33	7.58	8.80	8.67	9.00	9.43	13.50	9.78	30.86	56.00	73.00	78.00
		10.00	10.00	7.67	6.92	9.60	9.67	10.00	10.00	14.75	10.00	31.67	90.00	84.00	91.00
	10.00	6.00	5.00	9.00	7.50	9.60	9.33	9.67	9.14	14.13	8.89	30.51	83.00	71.00	83.00
	10.00	10.00	9.00	9.00	9.50	8.80	8.67	9.67	9.71	13.88	9.78	33.15	75.00	73.00	85.00
	10.00	7.33	10.00	8.00	8.83	9.20	9.50	9.33	6.86	12.94	6.89	28.66	86.00	72.00	83.00
	8.00	2.00	4.00			8.00	3.50								
Average	7.00	5.29	7.19	6.65	6.31	8.29	8.09	8.86	8.19	11.04	7.00	24.08	71.61	68.38	75.69
Std Dev	2.93	3.67	2.26	2.86	2.37	1.53	2.38	1.38	2.45	3.77	2.88	8.68	16.13	13.73	13.05

Appendix XI Students Grade for the PP Section in the Second Semester 2014 - 2015

Quiz 1	Quiz 2	Quiz 3	Quiz 4	Quizzes Total	Assignment 1	Assignment 2	Assignment 3	Assignment 4	Assignments Total	Practical	Lab Total	Midterm	Final	Total Average
6.00		5.00		3.00	7.00	6.00			5.00		8.00			
10.00	6.00	6.00	4.00	7.00	8.00		9.00		7.00	4.00	18.00	58.00	62.00	60.00
6.00	8.00	7.00	3.00	6.00	7.00		8.00		6.00	5.00	17.00	71.00	76.00	65.00
5.00	10.00	5.00	4.00	6.00	7.00	6.00	9.00		8.00	6.00	20.00	73.00	74.00	75.00
7.00	6.00	7.00	8.00	7.00	9.00	9.00	9.00	10.00	13.00	7.00	27.00	85.00	94.00	90.00
5.00	6.00	8.00	7.00	7.00	8.00	7.00	9.00		8.00	7.00	22.00	73.00	77.00	74.00
4.00	5.00	3.00		3.00	9.00	7.00	7.00		8.00	4.00	15.00	44.00	20.00	55.00
9.50	7.00	6.00	4.00	7.00	10.00	9.00	9.00		10.00	5.00	22.00	59.00	68.00	73.00
4.00		7.00	7.00	5.00	3.00	7.00	10.00		8.00	5.00	18.00	49.00	51.00	55.00
6.00	7.00	7.00	8.00	7.00	9.00	9.00	9.00		10.00	8.00	25.00	74.00	77.00	80.00
7.00	8.00	8.00	7.00	8.00	10.00	9.00	8.00	6.00	12.00	7.00	27.00	72.00	76.00	82.00
5.00	7.00	6.00	5.00	6.00	8.00	8.00	9.00	8.00	12.00	6.00	24.00	58.00	61.00	67.00
5.00	5.00	4.00		4.00	4.00	3.00	7.00		5.00		9.00			
	7.00	7.00		4.00		7.00	10.00	7.00	9.00	5.00	18.00	60.00	65.00	65.00
5.00	4.00	4.00	4.00	4.00	3.00	5.00	7.00		6.00	3.00	13.00	51.00	48.00	55.00
5.00	9.00	7.00	5.00	7.00	9.00	6.00	8.00		9.00	4.00	20.00	66.00	51.00	63.00
7.00	10.00	7.00	6.00	8.00	10.00	7.00	8.00		8.00	6.00	22.00	48.00	67.00	64.00
4.00	4.00	6.00	4.00	5.00	6.00		8.00		6.00	4.00	15.00	40.00	39.00	55.00
4.00	4.00	4.00		3.00	6.00	7.00	8.00		8.00		11.00			
6.00	6.00	4.00	4.00	5.00	3.00		6.00		4.00	3.00	12.00	55.00	56.00	55.00
6.00	8.00	4.00	4.00	6.00	8.00	8.00	8.00	8.00	12.00	5.00	23.00	65.00	70.00	72.00
10.00	3.00	8.00		5.00	10.00	8.00	8.00		10.00	8.00	23.00	86.00	80.00	81.00

	10.00	9.00	8.00		7.00	8.00		9.00		7.00	4.00	18.00	73.00	66.00	65.00
	5.00				1.00	8.00	7.00			6.00		7.00			
	4.00	4.00	5.00		3.00	2.00		9.00		5.00	2.00	10.00	36.00	32.00	55.00
	8.00	9.00	7.00	5.00	7.00	7.00	8.00		8.00	9.00	7.00	23.00	74.00	75.00	75.00
	5.00	5.00	3.00	5.00	5.00	9.00	5.00	8.00	9.00	11.00	5.00	21.00	63.00	65.00	67.00
	4.00		3.00		2.00	4.00	5.00			3.00		5.00			
	7.00	9.00	6.00	4.00	7.00	9.00	8.00	8.00	5.00	11.00	5.00	23.00	55.00	70.00	68.00
Average	6.05	6.64	5.79	5.16	5.34	7.18	7.00	8.32	7.63	8.14	5.21	17.79	62.00	63.33	67.33
Std Dev	1.93	2.06	1.64	1.54	1.88	2.42	1.54	0.95	1.60	2.61	1.59	6.20	13.31	16.62	10.01

Appendix XII Students Grade for the Traditional Section in the First Semester 2014 - 2015

Quiz 1	Quiz 2	Quiz 3	Quiz 4	Quizzes Total	Assignment 1	Assignment 2	Assignment 3	Assignment 4	Assignments Total	Practical	Lab Total	Midterm	Final	Total Average
3.00	3.33			1.58	3.20		8.00	2.86	5.25		6.83	52.00	42.00	55.00
4.00	6.67	10.00	7.67	7.08	9.20		10.00	10.00	11.00	8.56	26.64	51.00	88.00	77.00
3.00	9.00	4.00	8.33	6.08	8.80	0.67	5.67	1.43	5.75		11.83	74.00	57.00	63.00
2.00		4.00										18.00		
	8.33	6.50	1.67	4.13		6.00	6.67	3.14	6.13	4.33	14.58	64.00	55.00	61.00
0.00	8.00	4.00	4.33	4.08	7.20	8.00	6.67	4.00	9.50	3.78	17.36	67.00	60.00	64.00
4.00	2.00	4.00				8.67								
6.00	6.67	7.50	9.00	7.29	6.40	9.17	8.33	9.71	12.81	8.33	28.44	77.00	69.00	80.00
4.00	7.00	4.50		3.88	8.00	9.33	7.00	2.86	9.88	2.67	16.42	60.00	58.00	55.00
10.00	10.00	9.00	9.67	9.67	7.60	9.17	10.00	9.43	13.69	9.44	32.80	74.00	80.00	86.00
5.00	2.00	6.50	2.33	3.96	8.00	10.00	8.00	10.00	13.63	8.67	26.25	64.00	68.00	73.00
3.00		4.00			6.40	7.83						30.00		
5.00	6.00	5.00	1.67	4.42	6.40	6.67	8.67	6.29	10.50	3.11	18.03	57.00	62.00	60.00
3.00	7.33	6.00		4.08	8.40	7.17		10.00	9.69	7.67	21.44	36.00	54.00	55.00
2.00	4.33	5.50	0.00	2.96	8.40	9.33	7.67	8.86	12.88	4.22	20.06	60.00	63.00	64.00
4.00	10.00	5.00	5.00	6.00	8.40	10.00	9.00	8.57	13.50	7.56	27.06	92.00	80.00	86.00
4.00	1.33	4.50	0.00	2.46			8.67	7.43	6.50	6.67	15.63	48.00	85.00	66.00
	0.00	4.00			5.60							46.00		
3.00	7.33	4.00	4.33	4.67	7.20	7.17	8.67	9.14	12.19	5.89	22.74	56.00	60.00	66.00
2.00	7.33	4.50	2.67	4.13	4.40	6.50	6.00				4.13	41.00		
10.00	7.67	9.50	9.67	9.21	10.00	9.17	9.00	9.43	14.06	9.56	32.83	73.00	85.00	87.00
3.00	1.00		4.33	2.08	8.80	5.67	9.33	4.86	10.50	7.00	19.58	56.00	57.00	61.00

	3.00	5.00	6.00	6.00	5.00	6.00	7.33	7.33	3.43	8.88	6.11	19.99	80.00	80.00	78.00
	3.00	2.67	4.00			3.20									
	3.00	5.33				9.60	10.00						27.00		
	4.00	7.67	9.75		5.35	6.00	8.17	8.00	7.43	11.19	8.44	24.99	77.00	82.00	81.00
	3.00		6.00	2.67	2.92		5.67	6.33		4.50	1.11	8.53	50.00	58.00	55.00
	8.00	6.00				8.80	10.00								
	3.00		4.00	2.67		4.80							70.00		
	4.00	5.33	7.00	5.67	5.50	8.00		9.00	9.14	9.88	6.44	21.82	53.00	66.00	68.00
Average	3.96	5.67	5.72	4.61	4.84	7.15	7.80	8.00	6.90	10.09	6.29	19.91	57.52	67.10	68.62
Std Dev	2.22	2.81	1.98	3.10	2.10	1.88	2.16	1.26	2.98	2.99	2.46	7.76	17.48	12.78	11.01

Appendix XIII Students Grades for the Traditional Section in the Second Semester 2014 - 2015

Quiz 1	Quiz 2	Quiz 3	Quiz 4	Quizzes Total	Assignment 1	Assignment 2	Assignment 3	Assignment 4	Assignments Total	Practical	Lab Total	Midterm	Final	Total Average
6.00		5.00	8.00	5.00		6.00	9.00		6.00	3.00	14.00	46.00	39.00	55.00
				0.00					0.00		0.00			
2.00		3.00		1.00			8.00		3.00		4.00			
6.00		6.00	5.00	4.00	9.00	8.00	8.00		9.00	6.00	19.00	73.00	69.00	65.00
3.00	3.00	4.00		3.00					0.00		3.00			
7.00	7.00		6.00	5.00	9.00	8.00	8.00		9.00	6.00	20.00	55.00	77.00	70.00
10.00	8.00	10.00	7.00	9.00	9.00	9.00	8.00	7.00	11.00	7.00	27.00	66.00	80.00	81.00
8.00	9.00	8.00		6.00	10.00	9.00	9.00	8.00	13.00	8.00	27.00	72.00	82.00	83.00
7.00	8.00	5.00	7.00	7.00	7.00	9.00	9.00		9.00	6.00	22.00	70.00	73.00	70.00
4.00				1.00					1.00		2.00			
6.00	6.00	3.00	5.00	5.00	9.00	7.00	7.00	9.00	12.00	4.00	21.00	63.00	82.00	66.00
5.00	4.00	4.00	4.00	4.00	9.00	9.00	9.00	7.00	12.00	3.00	19.00	60.00	61.00	62.00
7.00	10.00	10.00	4.00	8.00	9.00	7.00	9.00		9.00	8.00	25.00	69.00	86.00	84.00
7.00		3.00		3.00	5.00				3.00		6.00			
				0.00					1.00		1.00			
6.00	7.00	10.00	10.00	8.00	8.00	9.00	10.00	10.00	13.00	8.00	29.00	75.00	61.00	78.00
9.00	5.00			4.00	4.00	6.00	8.00		7.00		11.00			
6.00	6.00	10.00	5.00	7.00	8.00	9.00	10.00	5.00	12.00	6.00	25.00	72.00	81.00	75.00
6.00	4.00	7.00	4.00	5.00		3.00	5.00		4.00	3.00	12.00	52.00	46.00	55.00
8.00	8.00	6.00	7.00	7.00		9.00	9.00	10.00	10.00	6.00	23.00	70.00	77.00	73.00
5.00		5.00		3.00	5.00				3.00		6.00			
7.00	6.00		6.00	5.00	8.00	6.00			6.00	6.00	17.00	60.00	59.00	60.00

	4.00		4.00		2.00	5.00	8.00	8.00		8.00		10.00	50.00	81.00	60.00
	5.00	7.00			3.00	8.00	8.00	9.00		8.00		11.00			
	4.00				1.00					0.00		1.00			
	4.00	6.00	5.00	5.00	5.00	9.00	4.00	7.00		8.00	4.00	17.00	36.00	51.00	55.00
	3.00				1.00	4.00		8.00		5.00	1.00	7.00	52.00	36.00	55.00
	8.00		5.00		3.00	7.00	8.00	9.00		9.00	7.00	19.00	67.00	63.00	60.00
	6.00	7.00	5.00		5.00	9.00	8.00	9.00		10.00	4.00	19.00	71.00	72.00	65.00
Average	5.89	6.53	5.90	5.93	4.14	7.55	7.50	8.38	8.00	6.93	5.33	14.38	62.05	67.16	66.95
Std Dev	1.91	1.84	2.45	1.73	2.47	1.90	1.73	1.12	1.83	4.17	2.03	8.99	10.86	15.33	9.82