8-2016

An examination of the employment of the pair programming methodology as a collaborative instructional scaffold on college student procedural learning and programming self-beliefs

Ronald Erdei

Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Ronald Erdei

Entitled

An Examination of the Employment of the Pair Programming Methodology as a Collaborative Instructional Scaffold on College Student Procedural Learning and Programming Self-Beliefs.

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

John A. Springer Co-chair	James L. Mohler
David M. Whittinghill Co-chair	Shirl Donaldson

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): John A. Springer, David M. Whittinghill

Approved by: Kathryne A. Newton 7/25/2016

Head of the Departmental Graduate Program Date

AN EXAMINATION OF THE EMPLOYMENT OF THE PAIR PROGRAMMING METHODOLOGY AS A COLLABORATIVE INSTRUCTIONAL SCAFFOLD ON COLLEGE STUDENT PROCEDURAL LEARNING AND PROGRAMMING SELF-BELIEFS.

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Ronald Erdei

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2016

Purdue University

West Lafayette, Indiana

To my partner, Christine.

To my friend, Ferd.

To my father, Dad.

ACKNOWLEDGEMENTS

My students provided the inspiration for this dissertation, embraced the classroom changes that it precipitated, and trusted me with their thoughts, feelings, and perceptions. Without them, this dissertation would simply not have been possible.

Kyle Lutes, Guity Ravai, Alejandra Magana, and Brian Talbert all provided counsel and guidance during the course of this long and arduous journey.

John Springer allowed me to intellectually and academically roam. David Whittinghill made sure I never lost my way. Shirl Donaldson made sure I never lost perspective. And Jamie Mohler pointed me at really cool things of which I never would have otherwise known.

Christine made me smile.

Thank you all.

TABLE OF CONTENTS

Page

Page

GLOSSARY

Declarative (or conceptual) knowledge – understanding *what something is*. Schneider and Stern (2010) describe conceptual knowledge as "general and abstract knowledge of the core principles and their interrelations in a domain" (p.178).

Instructional scaffolding - a well-researched, commonly-practiced educational methodology founded in constructivist theory that advocates strong initial support for learners that decreases as a learner's competence increases (Sawyer, 2006).

Pair programming - a collaborative computer programming methodology in which two individuals, literally working side by side on a single computer, assume complimentary roles in the active pursuit of a programmatic solution (Beck and Andres, 2004).

Performance task – a learning activity in which learners perform action sequences and procedures to demonstrate their procedural knowledge. Performance tasks typically result in a tangible product or a physical performance (Kubiszyn & Borich, 2010).

Procedural knowledge – understanding *how to do something*. Schneider and Stern (2010) describe procedural knowledge as "knowledge of operators and the conditions under which they can be applied to reach certain goals" (p.178).

Triangulation - the use of multiple sources of data, multiple investigators, multiple theories, or multiple methods of collecting and analyzing data to "provide corroborating evidence" (Creswell, 2012, p. 251) or "confirm the emerging findings" (Merriam, 1998, p. 204).

User generalizability – a perspective on external validity that holds the user of a study responsible for determining the extent to which the findings apply to their specific situation (Merriam, 1998). Walker (1980) states that, “it is the reader who has to ask, what is there in this study that I can apply to my own situation, and what clearly does not apply?” (p.34).

ABSTRACT

Erdei, Ronald. Ph.D., Purdue University, August 2016. An Examination of the Employment of the Pair Programming Methodology as a Collaborative Instructional Scaffold on College Student Procedural Learning and Programming Self-Beliefs. Major Professors: John Springer, David Whittinghill.

Using a concurrent mixed methods case study approach, this study investigated the impact of employing the pair programming methodology as a collaborative instructional scaffold on student programming procedural knowledge and programming-related self-beliefs in an introductory computer programming course offered at a large university located in the Midwestern United States. Employing a design research theoretical perspective in a natural educational setting, the study used course performance data, survey data, and researcher observations to educe that employment of the pair programming methodology as a collaborative instructional scaffold facilitated a more efficient learning process as well as a learning process less reliant on instructors. However, employment of the scaffold did not facilitate any significant difference in amount of procedural knowledge ultimately learned by students. In essence: students learned faster and with less instructor assistance, but not more. Data was collected during a single semester of the course which had a final enrollment of 76 undergraduate students from science and technology disciplines. Analysis was primarily quantitative in nature, with qualitative data being quantified where possible. Findings were based on a cooperative learning theoretical framework, and results were analyzed to identify differential impact of the instructional scaffold by factors of interest to classroom practitioners.

CHAPTER 1: INTRODUCTION

This chapter introduces the research study. It identifies both the phenomenon investigated – employment of the pair programming methodology as a collaborative instructional scaffolding technique - as well as my reasons for wishing to investigate this phenomenon. This chapter also identifies the purpose and scope of the investigation, establishing significance by framing it within the broader body of research on pair programming and my prior research on pair programming.

1.1 Background

My students often report that learning to program is challenging. Current literature supports this anecdotal evidence, with two recent studies finding that nearly one in three students enrolled in a computer programming course fails to complete that course (Bennedsen and Casper, 2007; Watson and Li, 2014). Motivated both by a desire to assist and by intellectual curiosity, I wondered: what interventions were available to improve the process by which students learn to program?

One means to improve the learning process is through appropriate use of instructional scaffolding. Instructional scaffolding is a well-researched, commonly-practiced educational technique whereby support is temporarily provided as an individual learns (Sawyer, 2006). In computer programming, there are several instructional scaffolds available to instructors. The proposed study represents the fourth and final iteration in a series of design experiments

investigating the use of pair programming, a collaborative computer programming methodology readily employed as an instructional scaffolding technique, to improve the computer programming learning process in a natural educational setting.

1.2 Notable Findings of My Prior Research

I have completed three prior iterations of study on the pair programming collaborative scaffolding technique. The design of each new iteration reflects the findings of the prior iteration, as illustrated in Figure 1.1.

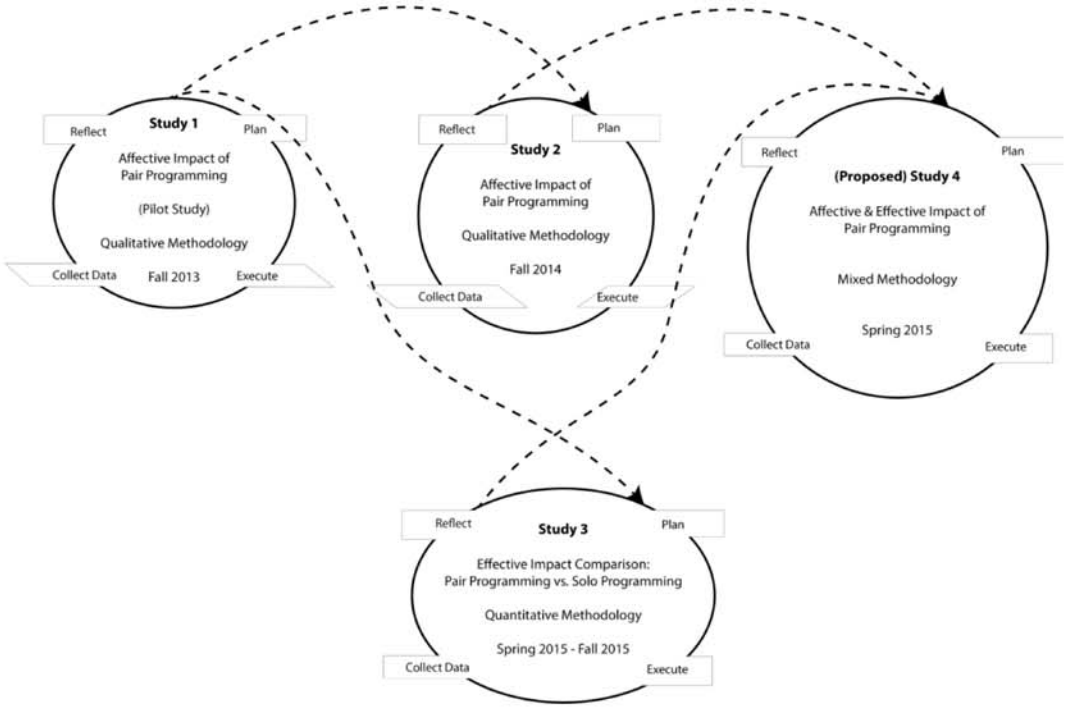


Figure 1.1. Iterations of the Investigation (including the proposed 4th study)

The first iteration of the investigation, in all respects a pilot study, was a single semester-long study conducted in a junior-level college computer programming course. The vast majority

of the participants (i.e., students) who opted to employ the scaffolding technique reported a high level of satisfaction with their decision, while the majority of those participants who opted to work individually report later regretting their decision and expressed desire for a peer with whom to collaborate upon challenging material (Erdei, Whittinghill, & Springer, 2014).

The findings of the second iteration of the investigation, conducted during a single semester-long study in a junior-level college computer programming course, both supported and extrapolated upon those of the prior iteration. Participants in the second iteration reported feeling both a connection to their peer collaborator, and a sense of responsibility to their peer collaborator. Specifically, participants reported not wishing to waste their peer's time nor cause them to receive a lower grade. Participants reported that as a result of this sense of responsibility they increased the amount of time preparing for class, invested time preparing for time spent working with their peer collaborator, and allocated additional time to completing assignments. Participants also reported feeling less dependent upon instructors for assistance, and a greater sense of enjoyment of the course than they had expected when starting the semester (Erdei, Whittinghill, & Springer, 2016).

The third iteration of the investigation, conducted during 2 sequential offerings of a junior level introductory programming course, employed a quasi-experimental quantitative design to evaluate the effect of the pair programming scaffolding technique on student summative assessment scores (i.e., computer laboratory exam grades) and course-related self-efficacy. While this iteration revealed no overall statistically significant difference on summative assessments or self-efficacy between the experimental group (i.e., the class employing the scaffolding technique) and the control group (i.e., the class that did not employ the scaffolding technique), this did not hold true when examined by gender. Traditionally a minority in the course, female students in the experimental group out-performed their peers in the control group on all summative assessments

throughout the semester. These performance differences were found to be statistically significant (Erdei, Whittinghill, Springer, & Magana, 2016).

1.3 Significance

There is a great deal of literature on the use of pair programming to improve learning in computer programming coursework. Like my prior research though, not all studies have found pair programming to improve learning. Regardless of findings, investigation into the use of pair programming to improve learning have overwhelmingly employed only quantitative analysis of classroom summative assessment (i.e., test scores). As such, these studies restrict learning to a single dimension – test performance – and measurement of learning to a single metric – test score. Similarly, investigations into the use of pair programming to improve learning have overwhelmingly limited their focus to declarative knowledge (i.e., knowing "what" something is, such as the definition of iteration) as compared to procedural knowledge (i.e., knowing "how" to do something, such as implement a loop programmatically). Even the rare study that takes "program quality" or "elapsed time" into consideration fails to do so from a learning science perspective, thus reflecting a general deficiency in the current body of literature: few investigations into the use of pair programming in the classroom have employed learning science principles. Most lack even a theoretical foundation to guide design and interpretation. I believed that by using both quantitative and qualitative methods of inquiry, focusing on procedural knowledge, and basing the investigation in learning science principles and theory, a richer understanding of this scaffold's impact on learning could be discerned.

Similarly, there is a great deal of literature advocating the use of pair programming to reduce the frustration and sense of isolation common among students learning to computer

program. However, there has been little attention paid to the underlying mechanisms leading to these benefits. I believed that further investigation into student self-beliefs would increase understanding of how pair programming reduces student discomfort when used as an instructional scaffold.

1.4 Statement of Purpose

The purpose of this concurrent mixed methods case study was to explore the impact of employing the pair programming methodology as instructional scaffolding in a natural educational setting using a design research theoretical perspective. Course performance data was used in this study as one means of evaluating the impact of the pair programming instructional scaffold on student learning, while observations of students as they program served as a supplemental second means of evaluating impact. Using data concurrently gathered via survey, an evaluation of the impact of the instructional scaffold on student self-beliefs complemented the evaluation of impact in this study.

1.5 Research Questions

The study aimed to answer the following research questions:

1. Was there a significant difference in learning between those students who employed the pair programming methodology as an instructional scaffold and those who did not?
2. Were there significant differences between the changes in programming-related self-beliefs undergone by those students who employed the pair programming methodology as an instructional scaffold and those who did not?

3. What differential impact of the instructional scaffold on learning was observed within factors of academic classification, academic discipline, gender, prior computer programming experience, and prior course (computer programming) performance?
4. What differential impact of the instructional scaffold on changing programming-related self-beliefs were observed within factors of academic classification, academic discipline, gender, prior computer programming experience, and prior course (computer programming) performance?
5. How do observations of students while programming help to explain any differences in learning between those students who employed the pair programming methodology as an instructional scaffold and those who did not?
6. How do observations of students while programming help to explain any differences in programming-related self-belief changes between those students who employed the pair programming methodology as an instructional scaffold and those who did not?

1.6 Assumptions

The following assumptions were inherent to this investigation:

1. Instructional scaffolding, when appropriately applied, is beneficial to the learning process.
2. There is a need to investigate the pair programming methodology as an instructional scaffold for computer programming courses.
3. Participants in this investigation attempted to do their best on course performance assessments, within the parameters of their specific educational goals.
4. Participants in this investigation were academically honest when completing the course performance assessments.

5. Participants in this investigation had access to all required course resources.
6. Participants in this investigation accurately and honestly answered questions upon the questionnaires.
7. The research methods employed in this investigation were appropriate to the research questions investigated and the natural educational setting in which the investigation occurred.

1.7 Limitations

The following limitations were inherent to this investigation:

1. This investigation was limited to students enrolled in CNIT 17500: Visual Programming, spring semester 2016, at the main campus of Purdue University.
2. This investigation was limited by the ability of participants to self-enroll in the CNIT 17500: Visual Programming laboratory section of their choosing.
3. This investigation was limited by an inability to control factors affecting participants outside of the classroom.
4. This investigation was limited by the willingness and ability of participants to collaborate with peers when employing the pair programming methodology.
5. This investigation was limited by the willingness and ability of participants to act naturally while being observed.
6. This investigation was limited by the researcher also being the course instructor.
7. This investigation was limited to the accuracy of the regular course instruments used in assessing mastery of CNIT 17500: Visual Programming learning objectives.
8. This investigation was limited to the accuracy of the Scott and Ghinea (2014) instrument assessing student self-beliefs.

9. This investigation was limited to the accuracy of the *Classroom Community* and *Group Processing* factors of the Summer, Gorin, et al (2005) instrument assessing the effects of collaborative group-learning.
10. The investigation was limited by the accuracy of the *Enjoyment Rating* within the Ryan and Connell (1989) Self-Regulated Learning (SRL-A) instrument.

1.8 Delimitations

The following delimitations were inherent to this investigation:

1. The Purdue University facilities in which the CNIT 17500: Visual Programming course components were hosted.
2. Students who dropped or withdrew from CNIT 17500: Visual Programming before data collection begins.
3. Participants who received academic accommodations as documented by Purdue University pursuant to the Rehabilitation Act of 1973 and the Americans with Disabilities Act of 1990.
4. The period of data collection, limited to spring 2016.

1.9 Summary

This chapter has provided the background to this investigation into the impact of employing a collaborative instructional scaffold. This chapter also discussed notable findings of my prior research on the topic, as well as significance of the investigation. The chapter also identified the purpose of the study and the research questions investigated during the study. Finally, this chapter identified assumptions, limitations, and delimitations of the study.

CHAPTER 2: REVIEW OF LITERATURE

There is a great deal of interest in the employment of the pair programming methodology, both in industrial settings and academic settings. As this study investigated usage of the pair programming methodology in an academic setting, this review of literature primarily focuses on research investigating the usage of this methodology in academic settings even though research on its usage in industrial settings does exist. However, particularly relevant research conducted in industrial settings will be included.

2.1 The Pair Programming Methodology

Pair programming is a collaborative computer programming methodology in which two individuals, literally working side by side on a single computer, assume complimentary roles in the active pursuit of a programmatic solution (Beck and Andres, 2004). The two roles in this programming methodology are that of *driver* and *navigator*, with one member of the team acting as the driver and one member acting as the navigator at any given time. The individual who has assumed the driver role controls the keyboard, typing the code (or creating the document) while focusing on the details of the program. The individual who has assumed the role of the navigator "actively observes the work of the driver, looking for tactical and strategic defects, thinking of alternatives, writing down 'things-to-do', and looking up references" (Arisholm, Gallis, Dyba, & Sjoberg, 2007, p. 65). In addition, the navigator serves as an "ever-ready brainstorming partner" (Nagappan, Williams, Ferzli, Wiebe, Yang, Miller, & Balik, 2003, p. 359). The two individuals periodically switch roles, repeatedly iterating between driver and navigator, so that each

individual spends equal time acting in each of the two roles. In addition to the writing of code, pair programming teams engage in other phases of the software development process, notably design and testing.

2.1.1 History

Though the term "pair programming" would not be coined for decades, the act of two programmers collaborating side-by-side has been practiced almost as long as the computer has existed. Fred Brooks, author of *The Mythical Man Month*, describes his experience pair programming less than a decade after the creation of the first electronic general purpose computer was created, "Fellow graduate student Bill Wright and I first tried pair programming when I was a grad student (1953-56). We produced 1500 lines of defect-free code; it ran correctly first try" (Williams, & Kessler, 2002, p. 8). Dick Gabriel, one of the developers of the Lisp programming language, describes pair programming usage as a "common practice at the M.I.T. Artificial Intelligence Laboratory" in the early 1970s (Williams, & Kessler, 2002, p. 11). Gabriel also credits pair programming as the methodology his team employed while developing Lisp. But it wasn't until Kent Beck developed the Extreme Programming methodology that the pair programming methodology was formalized (Beck, 2000). Research into the use of pair programming as a formal methodology in both academic and industrial settings begins at this time.

2.1.2 Benefits

The use of pair programming in college computer programming courses is common, as research suggests the employment of the methodology to be beneficial to students in multiple,

sometimes interconnected, ways. For purposes of this literature review, I have classified these benefits into four discrete categories: increased success in computer programming courses, increased learning in computer programming courses, reduced discomfort in computer programming courses, and increased student retention in computing majors.

2.1.2.1. Increased Success

Learning to program is generally accepted to be difficult for many students. Bergin and Reilly state, "It is well known in the Computer Science Education (CSE) community that students have difficulty with programming courses and this can result in high drop-out and failure rates" (p. 293). Bornat, Dehnadi, and Simon (2008) note, "Substantial failure rates plague introductory programming courses the world over, and have increased rather than decreased over the years" (p. 53). A worldwide survey of colleges and universities revealed only 67% of students in introductory programming courses pass the course – 33% either drop the course or fail (Bennedsen, & Caspersen, 2007). A follow-up study by Watson and Li (2014) found a worldwide pass rate of 67.7% for students in introductory programming courses, supporting the earlier findings.

Mcdowell, Werner, Bullock, Heather, and Fernald (2003) performed a broad investigation into the use pair programming over two semesters of an introductory programming course at the University of California. The investigation revealed "paired students were significantly more likely than non-paired students to complete the course, and consequently pass it" (p.602).

A similar investigation by Nagappan et al. (2003) into the use of pair programming in an introductory programming course was conducted at North Carolina State University. However,

this investigation was conducted in the service-course, taken by both computer science majors and non-majors. The results of this study indicate that the completion rate of non-computer science majors was improved by pair programming, however computer science majors did not see any significant improvement in regards to successfully completing the course. A related study by Nagappan et al. (2003) over three semesters included SAT score as a co-factor to be investigated. The results of this study indicate that for a given SAT score, students who pair programmed were more likely than those who did not to successfully pass the introductory programming course.

Increased success in introductory coursework is not universally observed, however. Somervell (2006) found no discernable difference in either successful completion rate or regular course performance metrics between students who pair programmed and those who did not in two parallel offerings of an introductory programming course. It should be noted that this study was comparatively small in scale though, being only a probe into the pedagogical approach.

2.1.2.2. Increased Learning

Whereas increased success reflects passing an introductory programming course, increased learning reflects increased mastery of the learning objectives for the course. This is typically measured via standard course performance assessments – such as homework assignments, quizzes, and examinations – in natural educational settings.

McDowell et al. (2003) found that those students who employed pair programming earned higher scores on programming assignments than those who did not pair. This difference was statistically significant. However, there was no discernable difference between those who paired and those who did not with regards to the course final examination, used by the researcher to assess the "extent to which students had mastered the course material" (p. 604).

Increased learning was suggested by the first of the Nagappan et al. (2003) studies, in which students who paired were observed to score higher on both means used by the researcher to assess mastery of course material, examinations and programming projects. However, no discernable difference was observed on either means of assessment in the second study. Taken together, this suggests that those students who pair program learn at least as much as those who do not.

Freeman, Jaeger, and Brougham (2003) had similar findings when investigating the use of pair programming in the first programming course required of engineering students at Northeastern University. Student performance on quizzes, final examination and overall course grade between those who paired and those who did not were not significantly different.

As previously mentioned, results were similar for Somervell (2006). There was no discernable difference between those who paired and those who did not in regards to regular course performance metrics.

2.1.2.3. Reduced Discomfort

Students in introductory programming courses often report feelings of anxiety (Wilfong, 2006). Decades ago this discomfort was recognized, as Gos states in 1996, "When the subjects of the study spoke of their fears of computers, or their past experiences with computers, they really were talking about bad feelings or experiences about programming" (p. 274). Furthermore, there is a high correlation between computer anxiety and decreased skills performance throughout coursework (Speier, Morris, & Briggs, 1995), but by "recognizing the prevalence of significant levels of anxiety amongst students learning programming, educators can structure learning and cognitive strategies to minimize this" (Connelly, Murphy, & Moore, 2009, p. 55). Freeman et al.

(2003) found pair programming to be effective at decreasing not only student anxiety, but the related discomfort frustration. Nagappan et al. (2003), who also found that students who paired experienced less frustration than those who did not, describing the lab conditions as such, "Solo lab sessions were quiet and appeared to be very frustrating for the students. Frequently, a student needed to wait 10-30 minutes to ask a question, often a fairly simple one. During this waiting period, or 'down time, students were often very unproductive (i.e., stuck)" (p.360). Nagappan et al. contrasts this with lab conditions in the paired lab, "paired labs were vocal and interactive. Students in paired labs engaged in extensive discussion throughout the entire lab session, and students seemed to help each other resolve questions" (p.360). Frustration and downtime were reduced, according to Nagappan et al., because pairs could often "piece together" the answer to questions and thus remain productive. Williams and Upchurch (2001) also report that students who pair programmed were "happier and less frustrated with the class" (p. 2) than the students who worked alone, while Braught, Eby, and Wahls (2008) also report findings indicating that students who pair programmed were less frustrated than those who worked alone.

Liebenberg, Mentz, and Breed (2012), focusing their investigation on secondary school girls' enjoyment of programming, had similar findings. Subjects prior to pair programming described the programming using the words "frustrating" and "stressful". However, after pair programming, the subjects repeatedly used the words "fun" and "more" to describe programming (e.g., "enjoyed it more", "much more fun", "more exciting") (p. 226).

2.1.2.4. Increasing Attractiveness of the Discipline to Women

An American Association of University Women (AAUW) report found that women are not avoiding high-tech careers due to failure, but instead because (a) there is a widely held belief

that a career in computing is neither well-rounded nor conducive to family life; (b) there is a widely held belief that a career in computing is conducted primarily in a competitive environment rather than a collaborative one; and (c) there is a widely held belief that computing occupations are solitary occupations lacking in social interaction. These findings reflect a general lack of "knowledge of the range of computing careers options available" (2000, p. 59).

The findings by Liebenberg et al. (2012) echo this report. Prior to pairing, subjects in the study had also reported strong feelings of socially isolation, causing them discomfort. The presence of a pair programming partner ameliorated this discomfort in subjects. Werner, Hanks, and McDowell (2004), who found that pair programming had a significant impact on female confidence in their work, advocate the use of pair programming as a means of increasing student retention, particularly among groups underrepresented in computing disciplines such as women. This recommendation is due to their belief that it "addresses several significant factors that limit" interest in computer science and computing occupations (p.1), a belief shared by The National Center for Women & Information Technology (NCWIT) who also recommend pair programming as a means of increasing female student retention (Jacobson, 2009).

2.1.3 Non-Academic Research

One investigation into pair programming usage in a non-academic, industrial setting is appropriate for inclusion here. Arisholm, Gallis, Dyba, and Sjoberg (2007) investigated *relative challenge of programming task* as a co-variant to pair programming. The study employed an experimental design in a controlled environment. The conceptual framework for the study incorporates programming methodology (pair, individual), programmer expertise, system complexity (an indicator of task difficulty), time and relative effort required of programmers, and

correctness of the solution developed by programmers. Findings of this study were that, in general, pair programming had no effect on elapsed time required to complete the programming task (duration) or correctness of the programming task. However, on tasks considered challenging relative to the programmer's skill level, pair programming usage showed a 48% increase in task correctness over those working alone. In contrast, on tasks considered simple relative to the programmer's skill level, only an increase in effort (actual person-hours) as compared to those working alone was observed. No benefit from pair programming usage was observed on tasks considered simple.

2.1.4 Deficiencies in the Literature

Deficiencies in the literature provide support for the importance of the study being proposed. One deficiency in the literature revealed by this survey is a lack of grounding in theory. Investigations almost exclusively focused on viability of pair programming as pedagogy with no interest in questions of "how" or "why". Relatedly, there are few investigations conducted from the perspective of learning science. As such, instruments employed to assess mastery of learning objectives in natural educational settings typically fail to identify the type of knowledge, declarative or procedural, being learned and assessed. Similarly, instruments and other assessment mechanisms employed in these studies rarely distinguish between mastery of high order learning objectives (evaluation, analysis) and low order learning objectives (understanding). This study was grounded in cooperative learning theory, employed a design research theoretical perspective, and employed a learning science approach and a focus on procedural knowledge with regard to instruments and interventions, thus avoiding the deficiencies revealed in this literature review.

2.2 Instructional Scaffolding

One of the foundations of this study was that pair programming was employed as a means of scaffolding. A review of literature on instructional scaffolding follows.

2.2.1 History

According to Quintana et al., scaffolding "has been traditionally defined as the process by which a teacher or more knowledgeable peer provides assistance that enables learners to succeed in problems that would otherwise be too difficult" (2004, p.338). This traditional definition can be seen even in one of the seminal works on scaffolding, Wood, Brunner, and Ross's investigation into the role of tutoring on problem solving abilities (1976) in which the term "scaffold" was coined. Wood, et al. held that the scaffolding process "enables a child or novice to solve a problem, carry out a task or achieve a goal which would be beyond his unassisted efforts" (p. 90) and focused on the actions of an expert, typically an adult, in "controlling those elements of the task that are initially beyond the learner's capacity, thus permitting" (p. 90) the learner to complete only the elements of the task that are within their ability.

More recent use of instructional scaffolding reflects a broadened view of scaffolds. No longer is the instrument of assistance limited to experts and adults. Instead, modern views of scaffolding hold techniques of instruction such as teacher-modeling, tools such as cue cards and software, and even peers of the learner as instructional scaffolds (Rosenshine & Meister, 1992). Miller (2009) identifies several modern instruments of scaffolding including clues, encouragement, explanations, modeling, prompts, and web links. When considering the breadth of modern scaffolds, Brush and Saye (2002) distinguish between what they refer to as soft scaffolds and hard scaffolds. Soft scaffolds are dynamic, situation-specific, and relatively reactive

in nature. The answering of ad-hoc questions by teachers, and the assistance of peers while learning to perform a task are two examples of soft scaffolds. In contrast, hard scaffolds are anticipatory in nature. The term "hard" is a reflection of the static nature of these scaffolds, which take the form of software-implemented support structures embedded within multimedia learning environments.

2.2.2 Theoretical Foundations

Instructional scaffolding is founded on the social constructivist theories of Vygotsky that posit knowledge is constructed when individuals engage with each other socially, through words and/or activities, about a shared problem or task (Merriam, 2007). One of the primary tenets of Vygotskian theory is the concept of a *zone of proximal development*:

> the distance between the actual developmental level as determined by independent problem-solving and the level of potential development as determined through problem-solving under adult guidance or in collaboration with more capable peers (Vygotsky, 1978, p. 86).

Figure 2.1 illustrates the zone of proximal development. Notice that the figure depicts 3 clearly delineated areas: the area containing tasks that the learner can perform without assistance (i.e., learners can do these things on their own), the area containing tasks that the learner can perform with assistance, but which otherwise would be unachievable (i.e., the zone of proximal development), and the area containing tasks that the learner is unable to perform even with assistance.

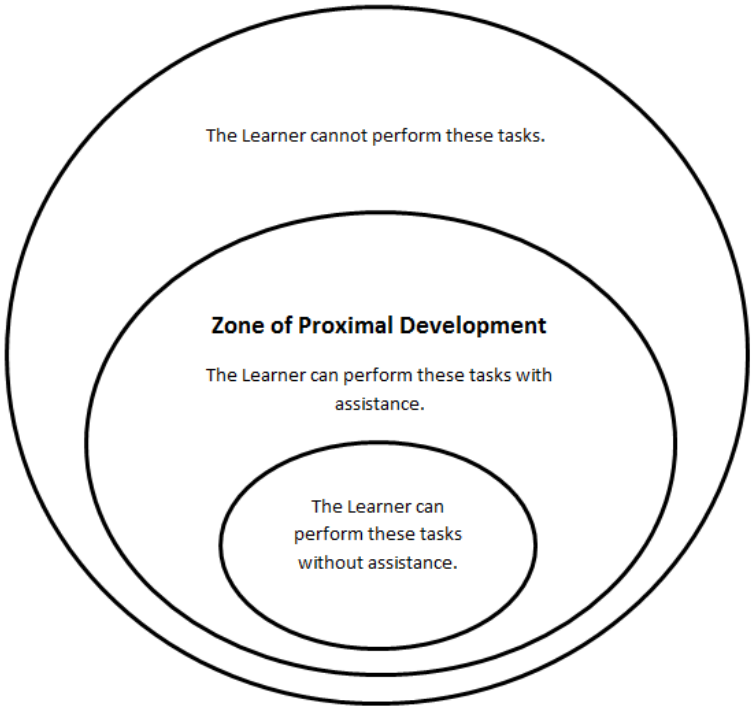


Figure 2.2. Zone of Proximal Development

When viewed through the perspective of Vygotskian theory, instructional scaffolds are the means by which assistance is provided to a learner thus allowing them to perform tasks which lie, at least initially, in their zone of proximal development. In this context, learning can be viewed as the processes allowing relocation of tasks between zones (Yelland & Masters, 2007).

2.2.3 Tenets

Over time, investigation into the use of instructional scaffolds has revealed several attributes common to their success (Beed, Hawkins, & Roller, 1991; Wood & Wood, 1996; Yelland & Masters, 2007). Those that were germane to this study are discussed.

2.2.3.1 Targeted and Dynamic

One tenet of effective scaffolding is that it be tailored to learner, task, learning environment. Yelland and Masters (2007) state that scaffolding "needs to be modified to suit circumstances of implementation (i.e., the scope of the task and the learner's own zone of development)" (p. 364), and "operate within the learner's zone of proximal development" (p. 364). Sawyer (2006) reinforces this tenet, stating that that in effective learning environments "scaffolding is gradually added, modified, and removed according to the needs of the learner" (p. 12). Palinscar and Brown (1984) note that amount of scaffold provided a learner should decreases as the learner becomes increasingly proficient. Because of this gradual reduction in learner assistance, this tenet of effective scaffolding is sometimes referred to as *fading* by researchers.

2.2.3.2 Temporary

Another tenet of effective scaffolding is that it be temporary (Tobias, 1982). In effective learning environments, scaffolding is eventually removed altogether (Sawyer, 2006) as the goal of is for the learner to internalize the knowledge required to perform the task, thus becoming independent of the scaffold (Beed, Hawkins, & Roller, 1991; Yelland & Masters, 2007).

2.2.3.3 Reciprocally Interactive

Yet another tenet of effective scaffolding is that it be reciprocally interactive (Delen, Liew, & Willson, 2014). Yelland and Masters refer to this tenet of effective scaffolding as collaborative (2007), while others refer to the social aspect of scaffolding. Quintana et al. (2004), in investigating the use of software in instructional scaffolding, identifies two common means by

which software is used to scaffold learning: "communicating processes to learners", and "eliciting articulation from learners to encourage reflection" (p. 338). Both are reciprocally communicative in nature. When learners articulate and externalize their developing knowledge, they learn more effectively (Bransford, Brown, & Cocking, 2000). As pointed out by Sawyer (2006) "articulating and learning go hand in hand, in a mutually reinforcing feedback loop" during which "the best learning takes place when learners articulate their unformed and still developing understanding" (p. 12). Articulation, when partnered with reflection, is considered the key to learning in many pedagogical approaches, not simply scaffolding (Quintana, et al., 2004). For example: when developing her scaffolded knowledge integration framework, Linn (1995) incorporated bi-directional reflection and articulation among group members as one of the primary means of "making thinking visible" (p. 106).

2.2.3.4 Learning Needs To Occur Prior To Task Completion

The last tenet of effective scaffolding discussed in this literature review is that learning needs to occur prior to completion of the scaffolded task. Wood, Brunner, and Ross (1976) stated that "comprehension of the solution must precede production" lest the learner be unable to "produce the steps leading to (the solution) without assistance" (p. 90). Thus the learner remains dependent upon the scaffold, instead of becoming independent of it. Pea (2004) in particular voices concern regarding this tenet, drawing attention not only to ineffective scaffolds that fail to assist in learning, but also to scaffolding that enables learning of the wrong task. To illustrate this point Pea refers to Clever Hans, a horse thought to have been taught arithmetic by his owner, a mathematics teacher, in the early 1900s. Unknown to the mathematics teacher, Clever Hans was reacting to the body language of the humans around him. The mathematics teacher had thus

inadvertently scaffolded learning of the wrong task - recognition of human body language – allowing the correct answer to be discerned, just not through understanding.

2.3 Summary

This chapter has identified existing literature relevant to this investigation into the impact of employing a collaborative instructional scaffold. Specifically, this chapter discussed the history of pair programming and its benefits in an academic setting. This chapter also identifies deficiencies in the body of literature on pair programming in an academic setting. Finally, this chapter discusses the history, tenets, and theoretical underpinnings of instructional scaffolding.

CHAPTER 3: APPROACH, FRAMEWORK AND METHODOLOGY

This chapter details the methods that were used in the study. This includes the research design, research ideology, theoretical framework, study environment, case description, researcher access, data collection procedures, and data analysis procedures. A discussion of internal validity, external validity, and proposed validation strategies concludes this chapter.

3.1 Research Design

As is often the case in natural educational settings, both the problem investigated and the environment in which the study occurred are complex. Creswell states that often "the use of either qualitative or quantitative approaches by themselves is inadequate to address the complexity" (2008, p. 203) faced by social, learning, and health science researchers. In such cases, Creswell advocates the use of a mixed methods approach as it allows greater insight to be gained via the combined use of qualitative and quantitative approaches than either approach singly. A mixed method research approach was thus selected for the study.

Several mixed method research designs were investigated prior to determining the design used in the study: the case study research design. The case study research design is a fundamental design common in both qualitative and mixed methods research. It was selected for several reasons, first and foremost because it is a "design particularly suited to situations where it is impossible to separate the phenomenon's variables from their context" (Yin, 1984). Because of this researchers seeking holistic explanation of a phenomena, an aim described by Cronbach

(1975, p. 123) as "interpretation in context," often employ the case study research design. Merriam (1998) elaborates on this aspect of the case study design, stating that "interest is in process rather than outcomes, in context rather than specific variables, in discovery rather than confirmation" (p. 19). Additionally, the case study research design allows a great deal of flexibility in regards to the methods of data collection employed. As Merriam (1988) points out, "any and all methods of gathering data from testing to interviewing can be used in a case study" (p. 10). Finally, the well-defined boundaries of a college course align well with the need to identify boundaries of the case, the "single unit" or "bounded system" to be investigated (Merriam, 1998, p. 19).

Though qualitative research is generally considered inductive in nature, case studies may employ both inductive and deductive reasoning, particularly if they employ mixed method data collection strategies or survey data collection strategies. One particular strategy, referred to by Creswell (2008) as the "Concurrent Transformative Strategy" (p. 215), is a mixed method research approach employing concurrent qualitative and quantitative data collection in which the researcher is guided by a specific theoretical perspective. In the case of this study, the guiding perspective was that of design research. This theoretical perspective is, as Creswell points out, evident in the purpose and research questions of the study, and serves as the "driving force behind all methodological decisions" (p. 215).

3.2 Theoretical Perspective

The study employed a *design research* approach. Described as an "emerging paradigm for the study of learning in context through the systematic study of instructional strategies and tools" (DBRC, 2003, p. 5), design studies are located in the larger field of learning sciences,

having evolved from the constructivist philosophies of Piaget, Vygotsky, and Dewey (Confrey, 2006). Also known as design studies, design experiments, and design-based research methods, design research is a methodology employed by researchers in the learning sciences seeking to iteratively design, test, and refine educational processes, typically in natural educational settings (Collins, Joseph & Bielaczyc, 2004).

Unlike many research methodologies, design research does not maintain the assumption that "research is contaminated by the external influence of the researcher" (Wang & Hannafin, 2005, p. 6). Instead "researchers manage research processes in collaboration with participants, design and implement interventions systematically to refine and improve initial designs, and ultimately seek to advance both pragmatic and theoretical aims affecting practice." This research approach was developed to address "theoretical questions about the nature of learning in context," that is, in a real-world setting as compared to a laboratory (Collins, Joseph & Bielaczyc, 2004, p. 16). The focus on learning in context is a hallmark of the design research methodology and is the means by which researchers employing the methodology seek to address a widely recognized problem in the learning sciences, that "educational research is often divorced from the problems and issues of everyday practice" (DBRC, 2006, p. 5). Described by the National Research Council (2002) as a "sharp divide between education research … and the practice of education in schools" (p. 14), this divorce of research from practice has prevented researchers and practitioners from developing the "kinds of cross fertilization that are necessary in fields where research and practice should develop reciprocally" (p. 15). Design researchers however embrace this reciprocity, rejecting the view that "pure research is conducted in a laboratory or experimental setting," with the results later being exported to a classroom setting (Confrey, 2006, p. 136). Instead, design researchers conduct research within the complexity of the classroom, where applied research and pure research are intertwined.

3.3 Theoretical Framework

Given the nature of the research treatment – employment of a collaborative instructional scaffolding technique – the study employed the theories of cooperative learning as its theoretical foundation. Cooperative learning is the "instructional use of small groups" to enable students working together to "maximize their own and each other's learning" (Johnson & Johnson, 2008, p.786).

According to Slavin (1987), there are two major theoretical perspectives forming the foundation of cooperative learning: developmental and motivational. The developmental perspective of cooperative learning is based in the theories of constructivists, Piaget and Vygotsky. It focuses upon the quality of interaction amongst students, advocating that exposure of individuals within a group to the higher-quality thinking of peers will precipitate higher-quality thinking in turn. According to the developmental perspective of cooperative learning, students will "learn from one another because in their discussions of the content, cognitive conflicts will arise, inadequate reasoning will be exposed … and higher quality understandings will emerge" (p. 1162). The second theoretical perspective identified by Slavin as foundational of cooperative learning, the motivational perspective, focuses on the "reward or goal structures under which group members operate" (p. 1162). According to the motivational perspective of cooperative learning, rewards should be attained through group performance, not individual performance, thus providing students an incentive to assist their peers so that the group succeeds.

Johnson and Johnson agree with Slavin regarding the developmental perspective of cooperative learning. Like Slavin's developmental perspective, Johnson and Johnson's (2008) developmental perspective focuses upon high-quality interaction among individuals during which knowledge is socially constructed, specifically from "cooperative efforts to learn, understand, and solve problems" (p. 789). However, instead of a single overarching motivational perspective on

cooperative learning as Slaving posits, Johnson and Johnson instead identify two related cooperative learning perspectives on motivation: the behavioral learning perspective and the social interdependence perspective. The behavioral perspective, based on the works of Skinner, Bandura, and other behaviorists, focuses upon the impact of group rewards and group reinforcers on learning while the social interdependence perspective focuses on the interdependence of group members. According to Johnson and Johnson, social interdependence exists "when individuals share common goals and each person's success is affected by the actions of others" (p. 789). Relatedly, when group members promote the success of each other, promotive interaction occurs between group members. Promotive interaction can take many forms, with examples including: providing/receiving assistance, exchanging resources or information, giving/receiving feedback on group-related tasks, and engaging in the interpersonal interaction needed for a group to function well. Of Johnson and Johnson's three perspectives on cooperative learning, it is this social interdependence perspective that has elicited the most interest from researchers. As conceived by Johnson and Johnson, effort to achieve, positive relationships with group members, and social competence all contribute to promotive interaction, which in turn leads to positive interdependence among group members.

3.4 Study Environment

The investigation was conducted at the main campus of a large land-grant university located in the Midwest United States: Purdue University. The university offers more than 200 majors to students, within its 10 colleges (Purdue Majors and Minors, n.d.). According to the Purdue University Office of Enrollment Management (*Purdue University West Lafayette Enrollment Summary: Fall 2015*, 2015), total enrollment for the main campus Fall 2015 was 39, 409 students, 29,497 (74.8%) of whom were undergraduates. The vast majority of these

undergraduate students were classified as full-time (95.4%). Among undergraduate students, the average age was 20.5 years old, males outnumber females nearly 3 to 2 (57.3% to 42.7%), and 2,568 (8.7%) identified as members of underrepresented minorities. The 4 largest colleges by undergraduate enrollment were: College of Engineering (7,928 undergraduate students), College of Health and Human Science (3,910 undergraduate students), College of Science (3,589 undergraduate students), and College of Technology (3,313 undergraduate students).

3.5 Case Identification

The case was a formal examination of the impact of employing a collaborative instructional scaffolding technique in the laboratory component of CNIT 17500: Visual Programming. Creswell (2012) indicates a key to case study research is sufficiently defining the case, including the identification of boundaries and parameters such as time and place. The following sections provide relevant details concerning the case to be studied. Since the treatment will occur in the laboratory component of the course, an exhaustive depth of detail regarding the course laboratory, its learning objectives, and the means by which these learning objectives are assessed has been provided.

3.5.1 Case Boundaries

The case was a single semester offering of CNIT 17500: Visual Programming. The course description provided to students via the Purdue University Course Catalog is:

> Credit Hours: 3.00. This course introduces event-driven application development and programming using a visual programming environment. Topics include problem solving

> and program design, control structures, objects and events, user interface construction, documentation, and program testing. Credit may be established in only one of: CPT 15500 or CPT 17500 or CPT 25000. PC literacy required. Typically offered Fall Spring Summer, 0.000 OR 3.000 Credit hours (Purdue University myPurdue Self-Service Catalog Entries: CNIT 17500, n.d.)

An introductory computer programming course offered by the Department of Computer and Information Technology at Purdue University, CNIT 17500 was typically taken as a selective by undergraduate students majoring in non-computing disciplines. The total course duration (including final examination) is 17 weeks, as is standard for courses offered at Purdue University during its spring and fall semesters.

3.5.2 Course Meetings

The course offering was held in the traditional lecture/laboratory style (i.e., "brick and mortar"), with all meetings being held on the Purdue University main campus. The meeting times and locations were scheduled by Purdue University as per its standard scheduling routine. The course had a single lecture option, being offered Monday and Friday afternoons from 1:30 pm until 2:20 pm. Thus, all students enrolled in the course attended the same lecture. The course had four laboratory meeting times available to students: Wednesday 9:30 am to 11:20 am, Wednesday 11:30 am to 1:20 pm, Wednesday 1:30 pm to 3:20 pm, and Wednesday 3:30 pm to 5:20 pm. Students could self-select which laboratory period they wished to attend, but had to attend the one selected throughout the entire semester. There was a single laboratory instructor who taught all four computer laboratories, thus all students had the same laboratory instructor on the same day of the week, albeit at potentially different times during the day.

3.5.3 Course Structure

The course was divided into two mandatory, complimentary components: lecture and laboratory. The course lecture met for 50 minutes, twice per week, in an appropriately sized lecture hall, and was used to highlight and reinforce concepts from the reading. In addition, lecture was used to demonstrate application of the concepts via coding demonstrations. As such, the lecture component of the course focused primarily on declarative knowledge at the lowest levels of Bloom's hierarchy: remembering (i.e., recall facts and basic concepts) and understanding (i.e., recognize and discuss ideas or concepts). The second component of the course, the laboratory component, met once per week in a computer laboratory for 110 minutes once per week. The computer laboratory contained all computer hardware and software necessary for students to create the computer programs used as formative assessments (i.e., programming assignments) and summative assessments (i.e., programming examinations) for the course. As the laboratory component of the course required students to apply concepts via the development of computer programs, it complemented the lecture component of the course by focusing on procedural knowledge at the mid-level of Bloom's Taxonomy: application (i.e., implement computer programs based on ideas and concepts), and analysis (i.e., implement computer programs containing multiple organized, differentiated code modules).

3.5.4 Course Learning Objectives

The course syllabus identified 28 learning objectives for students. A few of the learning objectives are purely conceptual, requiring students to merely remember or understand course content (i.e., declarative knowledge). In these cases, the learning objective was addressed only in lecture. However, the majority of course learning objectives were application oriented, requiring

students to apply concepts in code, analyze the applicability of various structures and techniques, and evaluate alternative means of success. These learning objectives were introduced in the laboratory component of the course through in-laboratory performance tasks, reinforced and formatively assessed via out-of-class programming assignments, and summatively assessed via in-laboratory programming examinations. Figure 3.4 depicts the learning objectives associated with each laboratory. Notice that, even though the semester was 17 weeks in duration, there were only 10 laboratories in which students focus on learning/mastering course learning objectives. The other weeks were comprised of summative assessment, administrative activities, preparatory activities, and spring break.

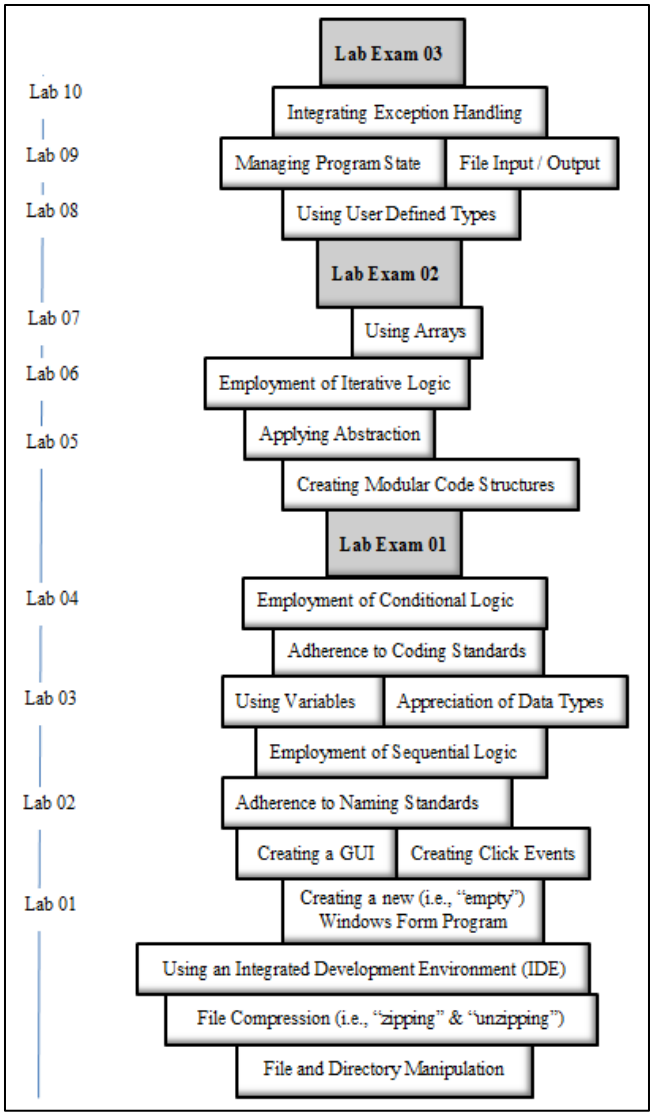


Figure 3.4. Learning Objectives by Laboratory.

The course had a standard process whereby it introduced, reinforced, and finally assessed students on learning objectives. The complete process is depicted in Figure 3.5, and applies to higher order learning objectives. It began with students being introduced to the learning objective via prescribed reading. An "open book" online quiz accompanied the prescribed reading, and was designed to help guide students as they complete the prescribed reading. Following this, the course instructor used one or more lecture periods to reinforce important facets of the learning

objective, elaborate on facets of the learning objective not sufficiently addressed in the reading, and/or demonstrate how the learning objective might be implemented in code. For lower-order learning objectives (i.e., those requiring only remembering and understanding on the part of the student, not application) the process was truncated prior to the In-Lab phase of the process.

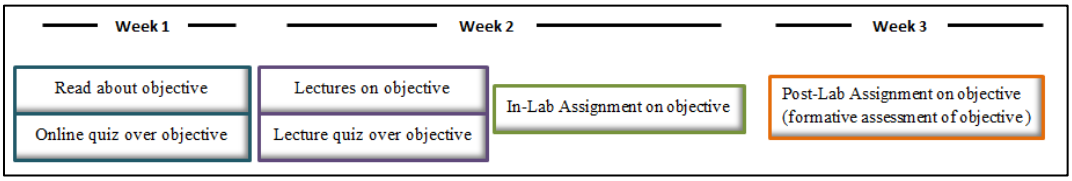


Figure 3.5. Standard Process used to Introduce, Guide, Reinforce, and Assess Students on Course Learning Objectives

3.5.5 Course Laboratory

The laboratory component of the course was mandatory for all students, and met for 110 minutes each week in a computer laboratory equipped with all hardware and software necessary for students to meet the computer programming learning objectives of the course. Each computer laboratory followed a standard process, as depicted in Figure 3.6. At the beginning of the laboratory, the laboratory instructor made course-related announcements, identified learning objectives and their location in the course textbook, and identified program specifications/requirements that students often overlook. The laboratory instructor may also have demonstrated how to implement some facet of the laboratory assignment, a scaffolding technique known as a *worked example*. Students then individually completed a performance task, typically the implementation of a computer program, targeting the specific learning objective(s) of that

laboratory. These performance tasks were referred to as “In-Lab” assignments. In-Lab assignments were designed to be completed during the laboratory period in which they were assigned, and to be relatively easy, serving as the first opportunity for students to actually write computer code satisfying that week’s learning objective.

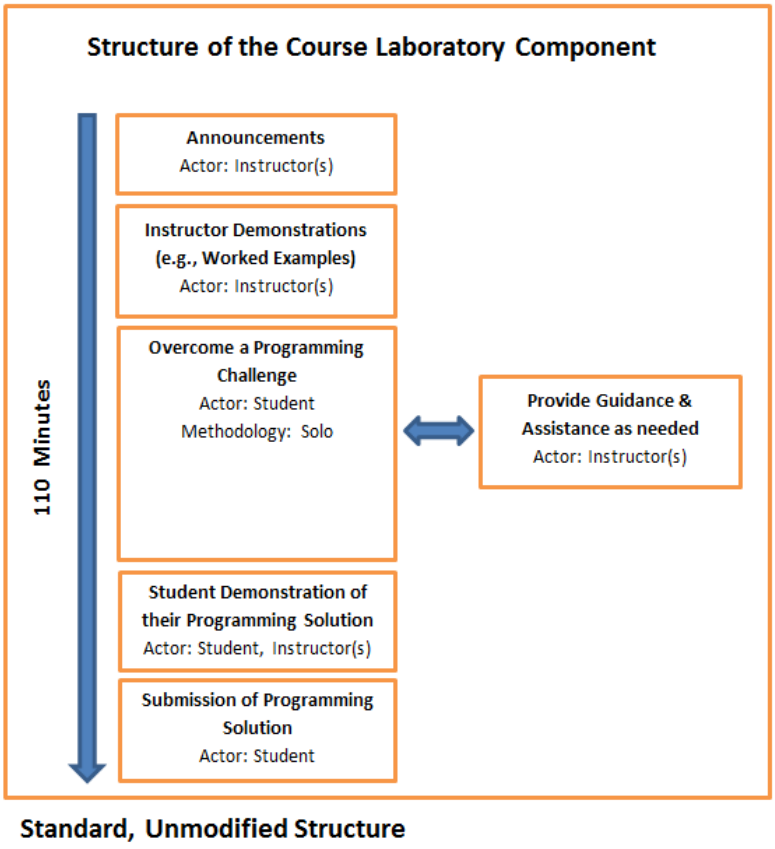


Figure 3.6. Structure of the Course Laboratory

While completing the In-Lab assignment, students had a breadth of resources that they may have used, which included, but are not limited to their textbook, any notes they created during lecture, and even online resources such as MSDN. In addition, students had access to the laboratory instructor who provided guidance to students requesting assistance while completing their In-Lab

assignment. When students had completed their In-Lab assignment, they were required to demonstrate it to the laboratory instructor, who confirmed that it met all requirements and "checked them off." The student then submitted the In-Lab assignment to Blackboard. However no additional assessment was performed for the In-Lab assignment – only the "check off."

As noted previously, a single laboratory instructor was assigned to the course. Normally, she alone would have taught the computer laboratories. However, after the first laboratory examination, I too attended the computer laboratories. The reason for this was twofold: first, to increase the number of instructors in the laboratory, and second to observe students while working on their in-lab programming assignments. Increasing the number of instructors had the benefit of decreasing the student to instructor ratio, thus allowing students more access to an instructor when they required assistance. Increasing the number of instructors in the laboratory also had the benefit of allowing one instructor to "check off" students who had completed the assignment, while the other remained available to provide assistance.

3.5.6 Participant Population

All students completing the course during the Spring 2016 semester were participants in the proposed study. The fact that the treatment was being evaluated was included in the course syllabus, and participants were informed of both the treatment and the study during the first course lecture. This allowed students who did not wish to participate in the study ample time to drop the course from their academic schedule.

3.6 Treatment

The treatment was deployed via a variation on what Carver (2006) refers to as a *now-and-later design*. A single class is split into two groups. One group receives the treatment while the second group serves as the control. After assessing the impact of the treatment versus the control, the control group then receives the treatment. This enables the researcher to "limit the impact of variables within the learning ecology by keeping a class together for all aspects of the intervention except for one manipulation" (p. 211). The now-and-later design is particularly attractive to educators, as all students eventually receive the intervention. Recall there were four laboratory sections to the course. Instead of splitting the class into two groups, as Carver describes, mid-semester students in two of the laboratory sections employed the pair programming methodology while students in the remaining two laboratory sections will continue to work individually. This treatment is depicted in Figure 3.7.

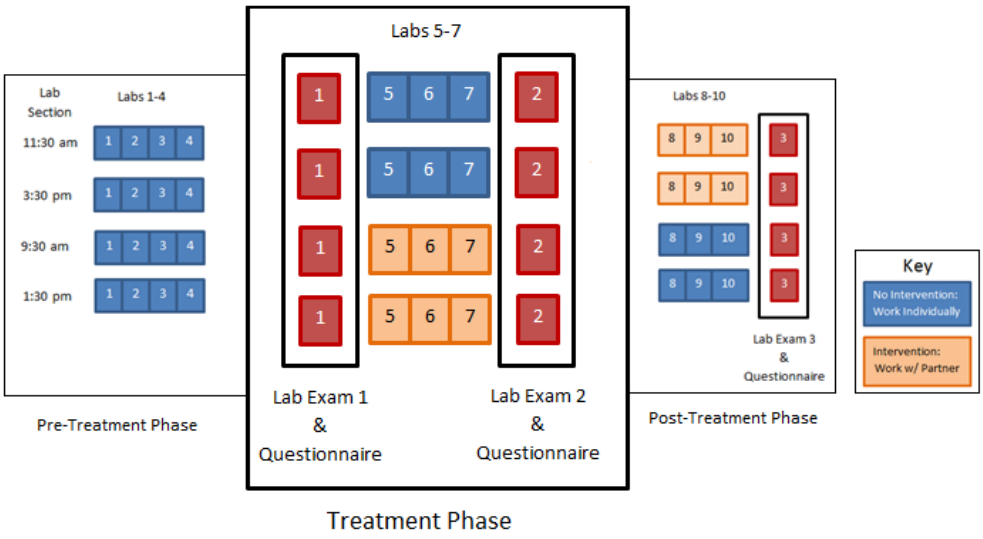


Figure 3.7. Proposed now-and-later design

The two laboratory sections that received the treatment (i.e.., in which students employing the pair programming methodology) were randomly determined. Each student in a laboratory section receiving the treatment was randomly assigned another student within the same laboratory section with whom to partner. Students remained partnered to the same individual for all three laboratories in which the treatment occurred. Figure 3.8 depicts the treatment, contrasting the programming methodology employed by the treatment group with that of the control group.

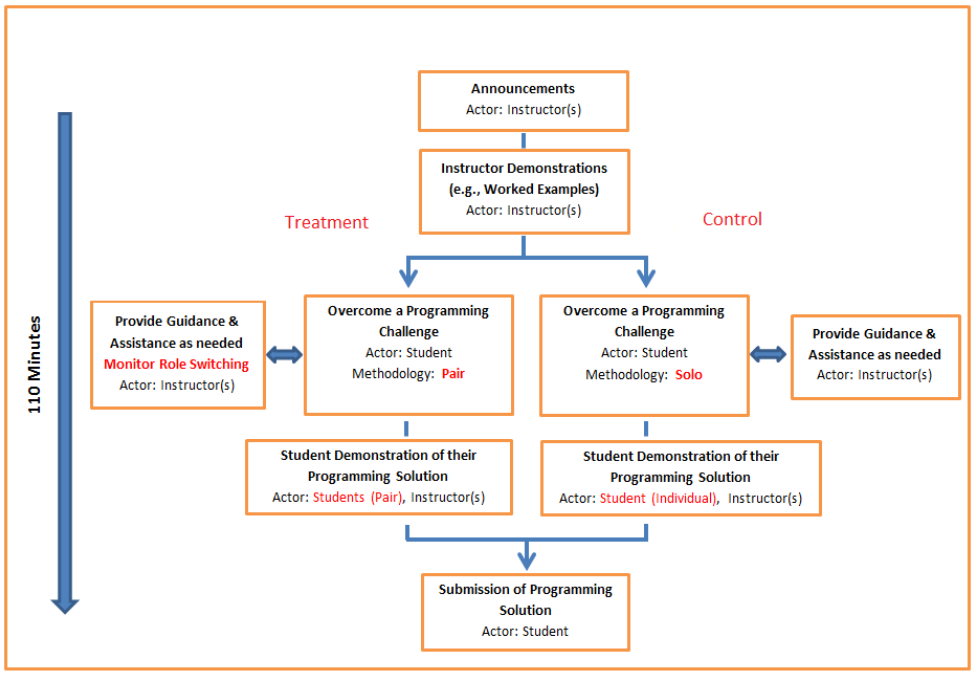
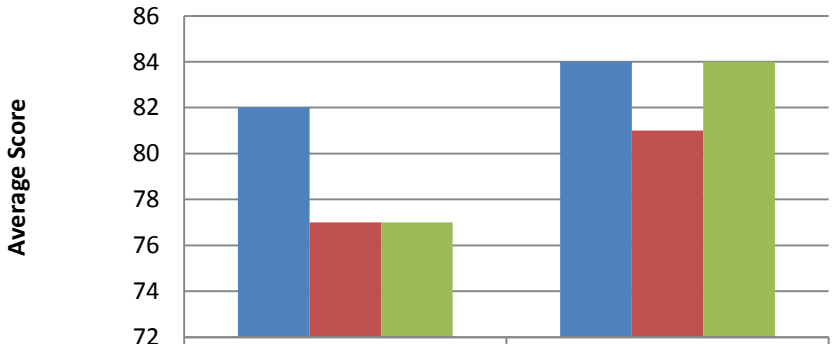


Figure 3.8. Comparison of Laboratory Structure for Treatment and Control Groups

The treatment was deployed mid-semester for two reasons. First, the learning objectives associated with the first four laboratories were relatively procedural in nature. As students were not conceptually challenged by these learning objectives, simply unfamiliar with how to

accomplish them, a collaborative instructional scaffolding technique was unlikely to help. Examples include: creating a directory with a specified name, compressing ("zipping") a program for submission, and creating a new Windows Form program in Visual Studio. The second reason for deploying the intervention mid-semester was that student performance historically drops on laboratory examination 2. Historically, it has the lowest average score of all three summative laboratory examinations, as shown in Figure 3.9. Anecdotal evidence (conversations with past students, end of semester feedback) suggests that the learning objectives associated with computer laboratories five, six, and seven to be the most challenging for many students. Anecdotal evidence also suggests that this is a very busy time of the semester for students, who may as a result invest less time in mastering the learning objectives of this course. In either case, additional instructional scaffolding had great potential to benefit the student learning process at this point in the course.



	Fall 2015	Fall 2014
Lab Exam 01	82	84
Lab Exam 02	77	81
Lab Exam 03	77	84

Figure 3.9. Historic Performance of Laboratory Examinations in CNIT 17500

Beven et al. (2002) and Williams and Kessler (2002) each provide guidelines for the use of pair programming in educational settings. During this study, pair programming occurred in a mandatory laboratory component of the course assuring student presence. Driver and navigators were physically present, side-by-side, working on a single University provided computer equipped with all necessary software. Drivers and navigators switched roles every 15 minutes. A kitchen timer with a loud alarm tracked time and provided notice when it was time to switch roles. Laboratory instructors enforced role switching, and roamed throughout the computer laboratory both providing assistance as needed and ensuring that navigators were performing the assigned functions of their role.

3.7 Data Collection

Creswell states "case study data collection involves a wide array of procedures as the researcher builds an in-depth picture of the case" (2012, p.162). Because of this, it was decided the study would include several vehicles of data collection, including: course performance data, questionnaire data, and observations. In addition, select academic/demographic information would be collected.

Data was not collected for students who failed to attend computer laboratories 5, 6, and 7 as they did not complete the In-Lab assignment (i.e., they did not receive either the treatment or the control).

3.7.1 Course Performance Data

Both summative and formative data was collected. Summative assessment has as its primary purpose the making of "a final success/failure decision" after completion of instructional activities (Popham, 2010, p. 10). In educational settings, tests and examinations often serve this function. Participant summative assessment data that was collected during the study included scores on the first two computer programming examinations (i.e., "lab exams") completed in the course. The first programming examination assessed participant mastery of learning objectives in the first four computer laboratories (i.e., labs 1-4), while the second programming examination assessed participant mastery of learning objectives in the subsequent three computer laboratories (i.e., lab 5-7). Both computer programming examinations were part of the standard course assessment process.

In contrast to summative assessment, formative assessment has as its primary purpose the improvement of incomplete instructional activities. Popham states "formative assessment is a planned process in which assessment-elicited evidence of students' status is used by teachers to adjust their ongoing instructional procedures or by students to adjust their current learning tactics" (2010, p. 270). Formative assessment data that was collected during this study included participant scores on three programming assignments, specifically the programming assignments associated with computer laboratories five, six, and seven (i.e., labs 5-7). All computer programming assignments were part of the standard course instructional and assessment process.

3.7.2 Questionnaire Data

Two questionnaires were made available to students after each computer programming examination. Both questionnaires will be administered via the course web site in Blackboard.

The first questionnaire was developed by Scott & Ghinea to evaluate self-beliefs among college students in computer programming courses (2014). The questionnaire contains nineteen Likert-type questions investigating five constructs: debugging self-efficacy, programming self-concept, programming interest, programming anxiety, and programming aptitude mindset. Before deployment, the questionnaire was adapted for use in the specific learning environment. Questions retained their original essence and context. All modifications were vetted by content experts to confirm changes were appropriate, and that the essence of the question remained unchanged. All nineteen questions were administered to all participants, regardless of receiving the treatment or not. The questionnaire as deployed can be found in Appendix A.

Scott & Ghinea (e.g. original)		Modified for use in our Study
DSE1	I am confident that I can understand Java exceptions (e.g., NullPointerException)	I am confident that I can understand Visual Basic exceptions (e.g., FormatException)

Figure 3.10. Instrument modification necessitated by learning environment differences

The second questionnaire was developed by Summer, Gorin, et al to evaluate perceptions of social connectedness, classroom community, and collaborative learning among college students (2005). The questionnaire contained twenty-six Likert-type questions investigating four constructs: social connectedness, classroom community, group processing: evaluation, and group processing: effect on individual. However, only three of the constructs were of interest in the proposed study (classroom community, group processing: evaluation, and group processing: effect on individual) while the fourth construct focused on a campus-level sense of social connectedness that is outside the focus of the proposed study. Therefore, only questions investigating the three constructs of interest were administered to participants. Five questions

were applicable to all students, and were administered to both those receiving the treatment and those not. An additional nine questions that investigated the two group processing constructs focus exclusively on a participant's current group, and thus were only applicable to those in the study receiving the treatment (i.e., that pair program). For example: "My group was successful in completing the requirements of most tasks." These nine questions were administered only to the group receiving the treatment (i.e., those pair programming). No modifications to the instrument needed to be made to accommodate learning environment in the proposed study.

Additionally, the second questionnaire included three questions adapted from Ryan and Connell (1989) that evaluates student enjoyment of classroom assignments. Again, before deployment the questions were adapted for use in the specific learning environment while retaining their original essence and context. All modifications were vetted by content experts to confirm changes are appropriate.

Finally, the second questionnaire included several original questions developed to evaluate dependence on the instructor for assistance, adherence to the pair programming methodology when paired, and partner dynamics when paired. Examples of the type of question developed include the following: "While completing the laboratory programming assignment, I required little assistance from the instructor," "When my partner was in the navigator role, they actively assisted me with the assignment," "My partner made me feel uncomfortable," and "I felt, at least in part, responsible for my laboratory partner's learning of the course material." The questionnaire as deployed can be found in Appendix B.

3.7.3 Observational Data

Observations for this study were made during computer laboratories 5-7. During these laboratories, I assisted the laboratory instructor by addressing student questions and checking off students once they had completed their In-Lab programming assignment. When not doing so, I moved around the computer laboratory free to observe students work and listen to their conversations.

Both the laboratory instructor and I recorded the number of questions posed by students, as well as the time at which each student completed their in-laboratory performance task (i.e., their In-Lab). In addition, personal observations were written in my notebook. Finally, pictures were periodically taken to provide visual record of participants working together and alone.

3.7.4 Demographic Data

The following academic/demographic data was collected for each participant: major, college/school, classification (i.e., senior, junior, sophomore, and freshman), gender, and prior programming experience. Major, college/school, and classification were collected from the course roster. Gender and prior programming experience were collected via questionnaire.

3.8 Data Analysis

The data analysis strategy employed in the study was designed with assistance from the Purdue University Statistical Consulting Service (PUSCS). Data analysis focused on data collected during the treatment phase of the study, and included course performance data,

questionnaire data, observation data, and demographic data. Data collected during the pre-treatment and post-treatment phase was not be analyzed.

3.8.1 Course Performance Data

Recall, the following course performance data collected was collected during the treatment phase of the proposed study: programming examination 1, programming examination 2, programming assignment 5, programming assignment 6, and programming assignment 7. The following strategy was followed for each examination and assignment: (1) normality of distribution was determined; (2) appropriate descriptive statistics were determined, be it mean and standard deviation or median, for both the treatment and the control data; (3) a two-sample test, be it t-test or Mann-Whitney U Test, appropriate to the normality of the distribution was performed to determine if the difference between treatment and control central tendency was significant. Where factors-specific sample size allowed, steps 2 and 3 were repeated to allow factor specific differential significance to be determined.

3.8.2 Questionnaire Data

Questionnaire data was first converted from the original Likert-type scale into a corresponding 5-point scale. After this, the analysis strategy for each instrument construct mirrored that employed for course performance data: (1) normality of distribution was determined; (2) appropriate descriptive statistics were determined, be it mean and standard deviation or median, for both the treatment and the control data; (3) a two-sample test, be it t-test or Mann-Whitney U Test, appropriate to the normality of the distribution was performed to determine if the difference between treatment and control central tendency was significant.

Where factors-specific sample size allowed, steps 2 and 3 were repeated to allow factor specific differential significance to be determined.

3.8.3 Observational Data

Observation data was quantified where appropriate, and then analyzed quantitatively. For example: number of questions asked by students to instructors during laboratory 5. The analysis will mirror the analysis process used for course performance and questionnaire data, as appropriate.

Observation data that was not quantifiable in nature was to transcribed to note cards. Unfortunately, very few non-quantifiable observations were made as students seeking assistance limited the opportunity for observation and proper note taking. However, had sufficient non-quantifiable observational data been gathered, it would have been analyzed via the Six-Sigma affinity diagramming process.

3.9 Validation

3.9.1 Internal Validity

Internal validity is the ability to accurately measure a phenomenon of interest. As such, internal validity in quantitative research typically focuses upon the instrument used to measure the phenomenon of interest. In contrast, internal validity in qualitative research typically focuses upon the process used to collect and analyze data when investigating the phenomenon of interest. As there is a breadth of methodologies employed in qualitative investigation, there is also a breadth of strategies (and nomenclature) aimed at ensuring the accuracy of qualitative

measurement and analysis. For this reason, Creswell (2012) recommends employing one or more accepted validation strategies to document the accuracy of the results. As this study was mixed methods (i.e., having both quantitative and qualitative elements) it followed the Creswell recommendation and employed several validation strategies to document the accuracy of the results. The specific validation strategies, selected for their alignment with the study's methodology, were: results of prior research studies informing the study's design, clarification of researcher bias, triangulation, peer review, member checking, and external audits.

3.9.1.1 Results of Prior Research Inform the Study Design

As noted in Chapter 1, this study was the 4th iteration in a series of related studies. Results of these prior studies informed the research questions of this study. Similarly, lessons learned during prior iterations informed both the treatment and data collection aspects of the study.

3.9.1.2 Clarification of Research Credentials and Bias

As Merriam (1988) points out, it is important to understand the position of the researcher within the study, as well as recognize any researcher bias or assumptions which may impact the study. Creswell (2012) also points out the importance of recognizing both relevant past experiences of the researcher and relevant orientations of the researcher, as both of which are likely to shape their interpretation of the data.

Over the past five years, I have taught thirteen undergraduate and mixed graduate/undergraduate computer programming courses at a large University. In that capacity, I

have been responsible for both facilitating and assessing the learning of 695 college students. Courses taught have all required the learning of both declarative knowledge (i.e., a conceptual component) and procedural knowledge (i.e., creating a computer program). In addition, I have authored several chapters in a college-level computer programming textbook, and presented 2 conference papers on the use of pair programming in the classroom. Last year, I earned both the Graduate Teaching Certificate from the Purdue University Center for Instructional Excellence and was awarded a Graduate Teaching Award from the Purdue University Teaching Academy. My curriculum vitae can be found in this document. I am committed to assisting my students succeed, not only in this course, but also at the University and in their subsequent professional life. University administered end-of-semester course/instructor evaluations repeatedly indicate that students recognize both my commitment and my ability to assist them. In approaching this multi-iteration study, my motivation was ultimately to assist my students in learning. This was the lens through which the scaffolding technique would be evaluated: impact on the student in our environment. I was hopeful that the overall benefit provided by the instructional scaffold would outweigh the overall cost incurred by its usage this case study. However, I was also cognizant of the possibility that this instructional scaffold would prove more costly to students than beneficial.

3.9.1.3 Triangulation

This study employed triangulation as a means of strengthening confidence in the accuracy of the findings. Triangulation is the use of multiple sources of data, multiple investigators, multiple theories, or multiple methods of collecting and analyzing data to "provide corroborating evidence" (Creswell, 2012, p. 251) or "confirm the emerging findings" (Merriam, 1998, p. 204). Mathison suggests viewing triangulation as a means of acquiring a "holistic

understanding of the specific situation" so as to better develop "plausible explanations about the phenomena being studied" (1988, p. 17).

3.9.1.4 Repeated Observations

This study employed a strategy of repeated observations to strengthen confidence in the accuracy of the findings. The course has four computer laboratory sections, all of which were observed for three computer laboratories (lab 5-7). As such, the study included approximately 24 hours of observation over a period of four weeks. Merriam identifies repeated observations as one means to "increase the validity of the findings" (1998, p. 204). Referred to as persistent observation by Creswell, this validation strategy incorporates "building trust with participants" and "learning the culture" to better enable both the identification of misinformation and the identification of "what is salient to study, relevant to the purpose of the study, and of interest for focus" (2012, pp. 250-251).

3.9.1.5 External Audits

This study employed external audits as a validation strategy. Creswell describes this validation strategy as the employment of an external agent - someone with "no connection to the study" - to examine "both the process and the product" of the study, verifying that the "findings, interpretations, and conclusions are supported by the data" (2012, p. 252).

The Purdue University Statistical Consulting Service (PUSCS) provided consultation on both the design and data analysis of this study. Once data had been collected, analyzed and the results interpreted, PUSCS was again consulted to audit the analysis process and results interpretation.

3.9.2 Reliability

Reliability is the ability to replicate research findings. Merriam states "reliability is problematic in the social sciences simply because human behavior is never static" (1998, p. 205). For this reason, Lincoln and Guba suggest instead thinking in terms of "dependability" or "consistency" of results obtained from data (1985, p.288). Creswell provides a complimentary view of reliability, stating it "often refers to the stability of responses to multiple coders of data sets" and suggests focusing on inter-coder agreement (2012, p. 253).

This study employed a single grader to evaluate (i.e., grade) all programming assignments and programming examinations. Because of this, inter-coder disagreement does not pose a threat to reliability of course performance data in the proposed study. To mitigate the threat of intra-coder disagreement (i.e., inconsistent grading by an individual), a skill-focused rubric with an analytic scoring system was developed for and used during grading of each computer programming assignment and examination. Skill-focused rubrics use demonstrated skills – not listed tasks – as evaluative criteria (Popham, 2010). Rubrics employing an analytic scoring system assign points on a "criterion-by-criterion basis" (p. 196), as compared to a holistic or collective basis.

3.9.3 External Validity

External validity is the ability to generalize findings. Merriam describes external validity as "the extent to which the findings of one study can be applied to other situations" (1998, p. 207), pointing out that – like reliability – external validity is often problematic in the social sciences. As such, several differing perspectives on external validity exist among qualitative researchers. Some qualitative researchers believe no generalization can occur, and accept this as a

limitation of the design. Other researchers reframe generalizability to better align with the design and focus of their study. As a mixed method study, this study will adopt what Merriam (1998) refers to as the *reader or user generalizability* perspective. This perspective of external validity allows the reader (or research user) to determine the extent to which the findings apply to their specific situation. Walker states that "it is the reader who has to ask, what is there in this study that I can apply to my own situation, and what clearly does not apply?" (1980, p.34). Firestone, who refers to this perspective as the *case-to-case transfer* perspective, describes it as occurring "whenever a person in one setting considers adopting a program or idea from another one" (1993, p. 17).

Another means of strengthening external validity in quantitative research is to use "standard sampling procedures," including random sampling, and then "treat the data quantitatively" (Merriam, 1998, p. 208). This technique was used in this study, and is another means by which external validity was strengthened.

3.10 Permissions

3.10.1 Course Instructor and Access

I was the instructor for CNIT 17500, the course in which the study occurred. As such, I was responsible for developing and administering all course assessments. All course assessments and questionnaires were submitted via Blackboard, and all student scores/grades maintained in Blackboard. As course instructor, I had access to the assignment and examination scores to be analyzed in the proposed study. Only data relevant to the study was downloaded from Blackboard for analysis, and all identifying information was scrubbed from records prior to analysis.

3.10.2 Human Subjects Approval

Human subject approval for the proposed study was granted for spring 2016 at Purdue University. Of import is that I was the course instructor seeking to investigate a novel instructional technique in the classroom, participants were students in the course all of whom participated and all of whom received the treatment, all activities took place in the classroom during regularly scheduled class time, participants did not receive any monetary compensation for their involvement, and participation in the study did not pose any additional risk to the subjects.

3.11 Summary

This chapter has identified the research design, theoretical perspective, and theoretical framework that guided this investigation into the impact of employing a collaborative instructional scaffold. This chapter also discussed the study environment and identified case boundaries, as well as the treatment employed in this investigation. Finally, this chapter identified data collection strategies, data analysis strategies, and validation strategies employed during this investigation.

CHAPTER 4: PRESENTATION OF THE DATA

As identified in prior chapters, the purpose of this concurrent mixed methods case study was to explore the impact of employing the pair programming methodology as instructional scaffolding in a natural educational setting. The research questions central to this study were (1) was there a significant difference in learning between those students who employed the pair programming methodology as an instructional scaffold and those who did not; (2) Were there significant differences between the changes in programming-related self-beliefs undergone by those students who employed the pair programming methodology as an instructional scaffold and those who did not; (3) what differential impact of the instructional scaffold on learning was observed within factors of academic classification, academic discipline, gender, prior computer programming experience, and prior course (computer programming) performance; (4) what differential impact of the instructional scaffold on changing programming-related self-beliefs were observed within factors of academic classification, academic discipline, gender, prior computer programming experience, and prior course (computer programming) performance; (5) how do observations of students while programming help to explain any differences in learning between those students who employed the pair programming methodology as an instructional scaffold and those who did not; (6) how do observations of students while programming help to explain any differences in programming-related self-belief changes between those students who employed the pair programming methodology as an instructional scaffold and those who did not? Course performance data, questionnaires, and repeated observations were used to educe the impact of employing the pair programming methodology as instructional scaffolding. As

discussed in prior chapters, observational data collection was limited in scope, with *time on task* and *number of questions posed to an instructor* being the data collected via observation.

This chapter presents data almost exclusively in quantitative aggregate. It begins by identifying software used in the analysis process. The chapter then verifies that there was no statistically significant difference between laboratory sections/ times on prior programming examination performance and subsequently identifies the treatment and control groups. It then characterizes the case according to factors of interest, participant demographics, and data collection schedule. Statistical comparison of the treatment and control groups is then undertaken for the overall case. Finally, comparison of the treatment and control groups is then undertaken for factor-specific subsets of the participant population where sample size allows.

4.1 Software Employed in the Analysis

Data from this study was stored and analyzed in an Oracle 11g Enterprise Edition database using statistical packages native to the platform. Testing for normality of distribution was performed using the SHAPIRO_WILKS variant of the NORMAL_DIST_FIT procedure located in the DBMS_STATS_FUNC package (DBMS_STAT_FUNCS, n.d.). The SHAPIRO_WILKS variant employs the statistical procedure developed by Shapiro and Wilk (1965) for testing a sample for normality. This variant was chosen because the Shapiro and Wilk test is widely considered "the best omnibus test for normality" (Rahman, & Govindarajulu, 1997). Analysis of variance (ANOVA) was performed using the STATS_ONE_WAY_ANOVA function (STATS_ONE_WAY_ANOVA, n.d.). T-tests comparing sample means were performed using the STATS_T_TEST_INDEP function (STATS_T_TEST_*, n.d.). Mann-

Whitney U tests comparing sample medians were performed using the STATS_MW_TEST function (STATS_MW_TEST, n.d.).

4.2 Treatment and Control Groups

As discussed in a previous chapter, the course had 4 laboratory sections available to students for self-enrollment. The laboratory sections met for 110 minutes each Wednesday throughout the semester. The location, facilities, content, and instructors were the same for all four sections. The first laboratory section began at 9:30 am, the second laboratory section began at 11:30 am, the third laboratory section began at 1:30 pm, and the fourth laboratory section began at 3:30 pm. A one-way ANOVA was conducted to compare the effect of laboratory section/time on student performance on the programming examination administered immediately prior to the treatment phase of the study (i.e., Programming Examination 1). The effect of laboratory section/time on programming examination performance was not observed to be significant at the $p < 0.05$ level for the four laboratory sections/times [$F(3, 72) = 2.576$, $p = 0.06$]. It should be the Shapiro Wilks test for distribution normality revealed the distribution of student performance not to be normal [$W = 0.907433$, $p = 0.00$]. However, as the sample sized exceeded 30 (N=76), "parametric tests" such as ANOVA "can be used on non-normal distributions without problem" (Pallin, 2010, p. 206). Figure 4.1 depicts this distribution.

The 9:30 am laboratory section and the 1:30 pm laboratory section were randomly determined to receive the treatment: the 45 students in these laboratory sections employed the pair programming methodology while completing their in-class programming tasks and activities. Students in the treatment group were randomly paired with another student enrolled in the same laboratory section. The 11:30 am laboratory section and the 3:30 pm laboratory section were

randomly determined to act as control: the 31students in these laboratory sections employed the traditional programming methodology (i.e., they worked individually) while completing their in-class programming tasks and activities.

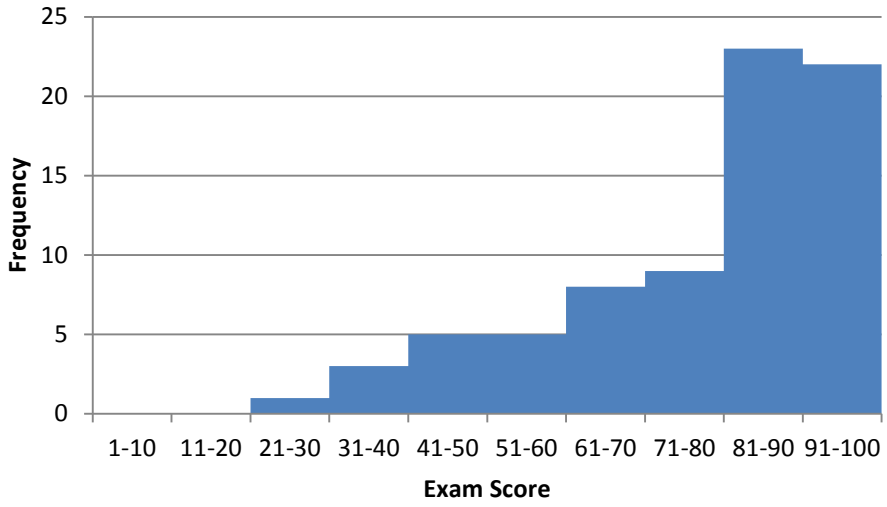


Figure 4.1. Distribution of Student Performance on Programming Exam 1.

4.3 Participant Demographics

Seventy-six students participated in the study. The demographic characteristics of these participants, in aggregate, are displayed in Table 4.1. Four additional participants withdrew from the course during the study: one participant was assigned to the treatment group while three were assigned to the control group. Performance data for these four participants is incomplete, thus their data have not been included in the study. One additional participant, while remaining enrolled in the course, failed to attend any of the course laboratories during which the treatment was applied and the study conducted. Thus this individual's data, limited as it is, have not been included. This individual was assigned to the control group.

Table 4.1.

Participant Demographics

	Treatment	Control	Total
Characteristic	N	N	N
Classification			76
Senior	15	8	23
Junior	12	15	27
Sophomore	18	7	25
Freshman	0	1	1
Discipline			76
Science	15	4	19
Technology	29	24	53
Other	1	3	4
Gender			76
Female	6	3	9
Male	39	28	67
Prior Examination Performance			76
High (80% - 100%)	27	19	46
Moderate (50% - 79%)	14	7	21
Low (0% - 49%)	4	5	9

Table 4.1 (continued).

Participant Demographics

	Treatment	Control	Total
Characteristic	N	N	N
Prior Programming Experience			76
Significant	2	1	3
Moderate	3	6	9
Slight	11	7	18
None	24	14	38
Undisclosed	5	3	8

4.4 Data Collection Schedule

Data was collected over a period of 5 weeks in the central portion of the Spring 2016 academic semester. Table 4.2 identifies the relative week and absolute date range for each observation, assignment, examination, and questionnaire used in the study. Spring Break occurred during Week 10 of the academic semester. No data was collected during Spring Break as the University was not in session.

Table 4.2.

Schedule for the Collection of Course Performance Data, Questionnaire Data, and Observation Data

Data Collected	Instructional Week	Date
Programming Examination 1	Week 7	February 24
Questionnaire: Self Beliefs 1	Week 7	February 25-29
Questionnaire: Connectedness 1	Week 7	February 25-29
Observation of Programming Activity 5	Week 8	March 2
Programming Assignment 5	Week 8	March 2-8
Observation of Programming Activity 6	Week 9	March 9
Programming Assignment 6	Week 9	March 9-22
Observation of Programming Activity 7	Week 11	March 23
Programming Assignment 7	Week 11	March 23-29
Programming Examination 2	Week 12	March 30
Questionnaire: Self Beliefs 2	Week 12	March 31-April 5
Questionnaire: Connectedness 2	Week 12	March 31-April 5

4.5 Comparative Analysis of Treatment and Control Groups

Parametric techniques, such as the t-test, make a great deal of assumptions regarding the populations from which samples are drawn. In contrast, non-parametric techniques lack “such

stringent assumptions, and are often more suitable techniques for smaller samples" (Pallin, 2010, p. 204). Fagerland (2012) recommends employing non-parametric tests when sample size is small, regardless of distribution shape. Because of this, the non-parametric Mann-Whitney U test was employed to determine significance of difference between treatment and control data. The Mann-Whitney U test is the non-parametric alternative to the t-test. It employs comparative logic focusing on sample median (Mdn) values instead of sample mean (M) values, as does the t-test. (Pallant, p. 227).

4.5.1 Overall Analysis of Treatment

4.5.1.1 Course Performance Data

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 60) and students who did not (Mdn = 59), Z= -.37, p = .711, r = -0.042. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 17) and students who did not (Mdn = 16), Z= -1.823, p = .068, r = -0.216. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 18.5) and students who did not (Mdn = 19), Z= -1.345, p = .179, r = -0.162. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who

received the treatment (Mdn = 16) and students who did not (Mdn = 16.5), Z= -1.099, p = .272, r = -0.128. These results are summarized in Table 4.3. Notice that N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.3.

Overall Student Performance (Score) on Standard Course Assessment Mechanisms

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	60	45	59	31	-0.37	0.711	-0.042
Programming Assignment 5	17	42	16	29	-1.823	0.068	-0.216
Programming Assignment 6	18.5	40	19	29	-1.345	0.179	-0.162
Programming Assignment 7	16	44	16.5	30	-1.099	0.272	-0.128

4.5.1.2 Observational Data

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did significantly differ between students who received the treatment (Mdn = 57 minutes) and students who did not (Mdn = 68

minutes), Z= -2.443, p = .015, r = -0.28. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did significantly differ between students who received the treatment (Mdn = 49 minutes) and students who did not (Mdn = 64 minutes), Z= -3.465, p = .001, r = -0.403. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 103 minutes) and students who did not (Mdn = 116 minutes), Z= -1.674, p = .272, r = -0.128. However, in-laboratory programming activity 7 was an intentionally long assignment. It was meant to challenge student's abilities to create programs efficiently. Only 68% of treatment group attendees (30 students out of 44) completed the activity, while only 55% of control group attendees (16 students out of 29) completed it. The results of the Mann-Whitney tests are summarized in Table 4.4, while median time on task is illustrated in Figure 4.2.

While completing in-laboratory programming activity 5, the 45 students in the treatment group posed 40 questions to instructors resulting in a ratio of 0.89 questions per student. In contrast, the 31 students in the control group posed 43 questions to instructors while completing the same programming activity, resulting in a higher ratio of 1.39 questions per student. While completing in-laboratory programming activity 6, the 45 students in the treatment group posed 37 questions to instructors resulting in a ratio of 0.82 questions per student. In contrast, the 31 students in the control group posed 79 questions to instructors while completing the same programming activity, resulting in a higher ratio of 2.55 questions per student. Finally, while completing in-laboratory programming activity 7, the 44 students in the treatment group posed 80 questions to instructors resulting in a ratio of 1.82 questions per student. In contrast, the 29 students in the control group posed 124 questions to instructors while completing the same programming activity, resulting in a much higher ratio of 4.28 questions per student. Figure 4.3 illustrates these ratios.

Table 4.4.

Overall Student Time on Task (Minutes) for In-Laboratory Programming Activities

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	57	45	68	31	-2.443	0.015	-0.28
Programming Activity 6	49	44	64	30	-3.465	0.001	-0.403
Programming Activity 7	103	30	116	16	-1.674	0.094	-0.247

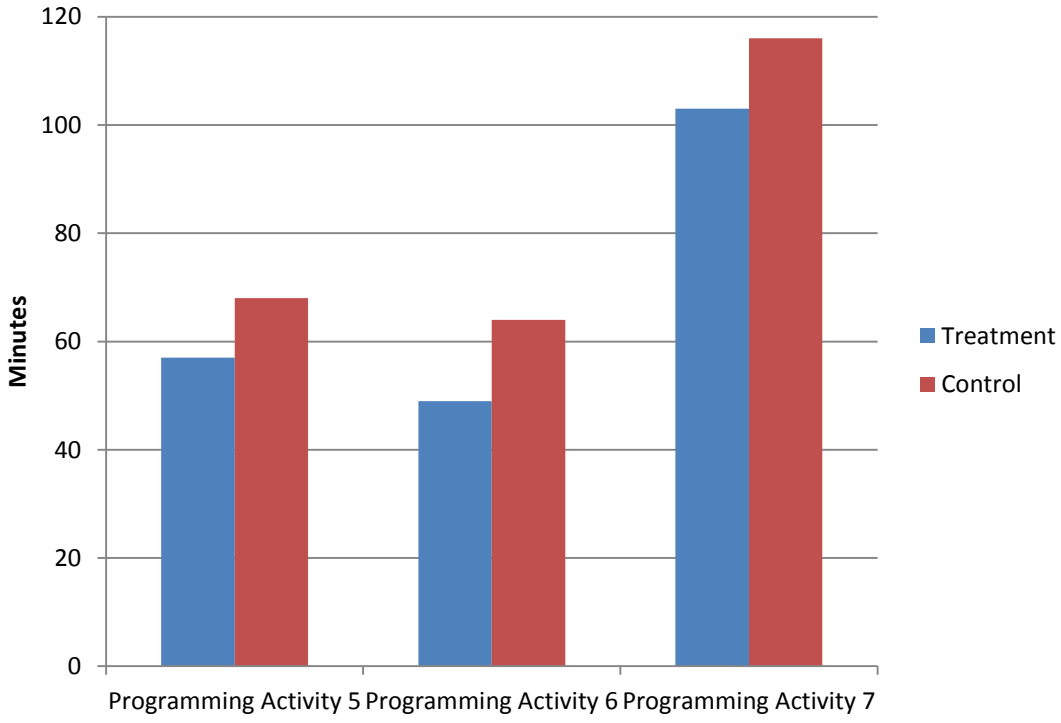


Figure 4.2. Median Time on Task for In-Laboratory Programming Activities

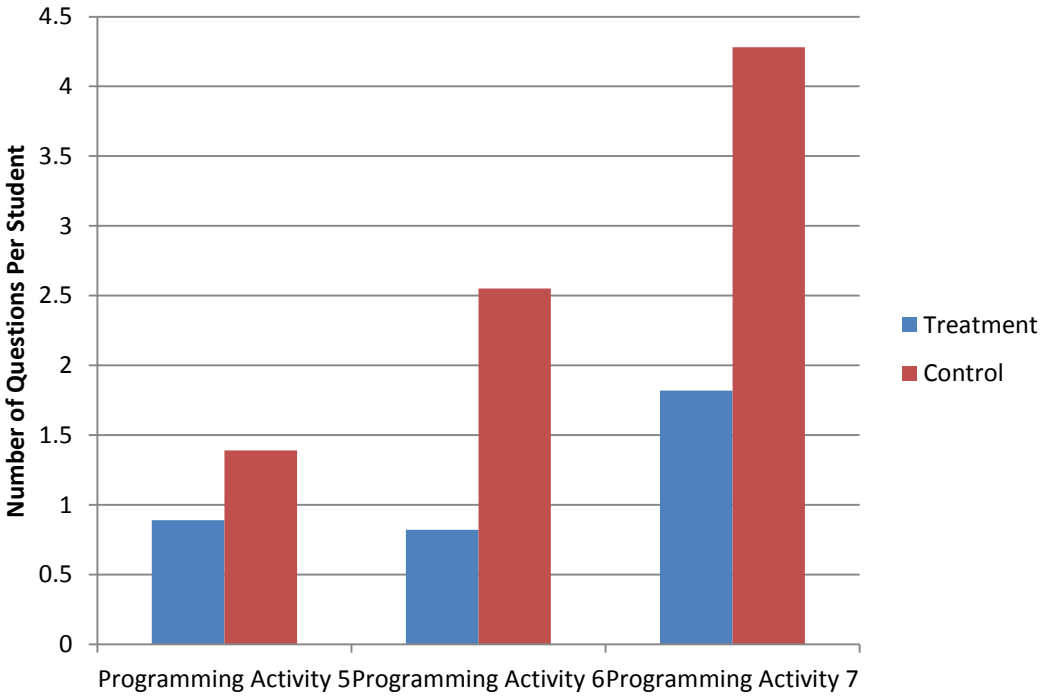


Figure 4.3. Number of Questions Per Student During In-Laboratory Programming Activities

4.5.1.3 Questionnaire Data

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = -0.17), Z= -1.595, p = .111, r = -0.195. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = 0), Z= -0.698, p = .485, r = -0.083. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0) and

students who did not (Mdn = 0.25), Z= -1.503, p = .133, r = -0.213. The Mann-Whitney test assessing change in the effect of group processing on the individual indicated that it did not differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = 0), Z= -0.282, p = .778, r = -0.033. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = -0.33), Z= -2.05, p = .04, r = -0.242. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = .13) and students who did not (Mdn = 0), Z= -1.404, p = .16, r = -0.195. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = -0.33), Z= -0.515, p = .606, r = -0.071. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = -0.13) and students who did not (Mdn = -0.25), Z= -0.23, p = .818, r = -0.032. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = 0), Z= -0.115, p = .908, r = -0.016. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.11) and students who did not (Mdn = -0.11), Z= -0.296, p = .767, r = -0.036. These results are summarized in Table 4.5.

Table 4.5.

Changes in Student Self-Beliefs during the Study

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	0.00	40	-0.17	27	-1.595	0.111	-0.195
Classroom Community	0.00	42	0.00	28	-0.698	0.485	-0.083
Debugging Self-Efficacy	0.00	31	0.25	19	-1.503	0.133	-0.213
Group Processing: Effect on Individual	0.00	43	0.00	28	-0.282	0.778	-0.033
Independence from/Dependence on Instructor	0.00	44	-0.33	28	-2.050	0.040	-0.242
Programming Anxiety	0.13	32	0.00	20	-1.404	0.160	-0.195
Programming Aptitude Mindset	0.00	33	-0.33	20	-0.515	0.606	-0.071
Programming Interest	-0.13	32	-0.25	20	-0.230	0.818	-0.032
Programming Self-Concept	0.00	33	0.00	19	-0.115	0.908	-0.016
Self-Efficacy	-0.11	42	-0.11	27	-0.296	0.767	-0.036

4.5.2 Analysis of Treatment by Classification

The data was analyzed to identify differential impact of the treatment on students classified as seniors, students classified as juniors, and students classified as sophomores. Unfortunately, analysis of differential impact of the treatment on students classified as freshman was not possible because only one student in the course was a freshman.

4.5.2.1 Students Classified as Seniors

Twenty-three students in the course were classified as seniors. Fifteen were members of the treatment group, while eight were members of the control group.

4.5.2.1.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 74) and students who did not (Mdn = 54), Z= -1.453, p = .146, r = -0.303. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 17) and students who did not (Mdn = 14), Z= -1.425, p = .154, r = -0.304. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 18) and students who did not (Mdn = 19.5), Z= -1.069, p = .285, r = -0.223. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who

received the treatment (Mdn = 17) and students who did not (Mdn = 16), Z= -0.288, p = .774, r = -0.061. These results are summarized in Table 4.6. Notice that N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.6.

Performance (Score) of Students Classified as Seniors on Standard Course Assessment Mechanisms

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	74	15	54	8	-1.453	0.146	-0.303
Programming Assignment 5	17	15	14	7	-1.425	0.154	-0.304
Programming Assignment 6	18	15	19.5	8	-1.069	0.285	-0.223
Programming Assignment 7	17	15	16	7	-0.288	0.774	-0.061

4.5.2.1.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did significantly differ between students who received the treatment (Mdn = 54 minutes) and students who did not (Mdn = 71

minutes), Z= -2.234, p = .025, r = -0.466. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did significantly differ between students who received the treatment (Mdn = 41 minutes) and students who did not (Mdn = 60 minutes), Z= -2.506, p = .012, r = -0.534. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 97 minutes) and students who did not (Mdn = 110 minutes), Z= -1.505, p = .132, r = -0.389. However, in-laboratory programming activity 7 was an intentionally long assignment. It was meant to challenge student's abilities to create programs efficiently. Only 73% of treatment group attendees (11 students out of 15) completed the activity, while only 67% of control group attendees (4 students out of 6) completed it. The results of the Mann-Whitney tests are summarized in Table 4.7, while median time on task is illustrated in Figure 4.4.

Table 4.7.

Time on Task (Minutes) for Students Classified as Seniors Completing In-Laboratory Programming Activities

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	54	15	71	8	-2.234	0.025	-0.466
Programming Activity 6	41	15	60	7	-2.506	0.012	-0.534
Programming Activity 7	97	11	110	4	-1.505	0.132	-0.389

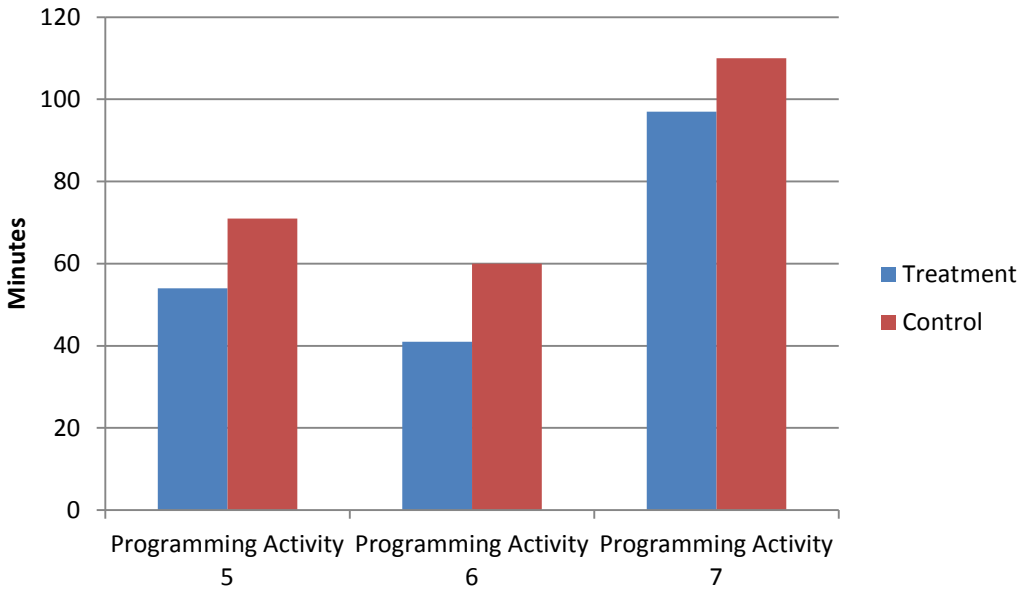


Figure 4.4. Median Time on Task for Students Classified as Seniors Completing In-Laboratory Programming Activities

4.5.2.1.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = -0.25), Z= -1.803, p = .071, r = -0.403. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = .13) and students who did not (Mdn = -0.25), Z= -0.673, p = .501, r = -0.15. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.25) and students who did not (Mdn = .5), Z= -1.818, p = .069, r = -0.548. The Mann-Whitney

test assessing change in the effect of group processing on the individual indicated that it did not differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = 0), Z= -0.707, p = .480, r = -0.158. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = -0.84), Z= -2.308, p = .021, r = -0.504. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = -0.13) and students who did not (Mdn = -0.75), Z= -0.518, p = .605, r = -0.144. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = -0.66), Z= -0.804, p = .422, r = -0.223. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = -0.13) and students who did not (Mdn = -0.5), Z= -0.896, p = .370, r = -0.248. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0) and students who did not (Mdn = 0), Z= 0, p = 1.000, r = .000. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.11) and students who did not (Mdn = -0.06), Z= -0.353, p = .724, r = -0.081. These results are summarized in Table 4.8.

Table 4.8.

Changes in the Self-Beliefs of Students Classified as Seniors during the Study

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	0.00	14	-0.25	6	-1.803	0.071	-0.403
Classroom Community	0.13	14	-0.25	6	-0.673	0.501	-0.15
Debugging Self-Efficacy	-0.25	9	0.50	2	-1.818	0.069	-0.548
Group Processing: Effect on Individual	0.00	14	0.00	6	-0.707	0.480	-0.158
Independence from/Dependence on Instructor	0.00	15	-0.84	6	-2.308	0.021	-0.504
Programming Anxiety	-0.13	10	-0.75	3	-0.518	0.605	-0.144
Programming Aptitude Mindset	0.00	10	-0.66	3	-0.804	0.422	-0.223
Programming Interest	-0.13	10	0.50	3	-0.896	0.370	-0.248
Programming Self-Concept	0.00	10	0.00	3	0	1	0
Self-Efficacy	-0.11	13	-0.06	6	-0.353	0.724	-0.081

4.5.2.2 Students Classified as Juniors

Twenty-seven students in the course were classified as juniors. Twelve were members of the treatment group, while fifteen were members of the control group.

4.5.2.2.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score and homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 62) and students who did not (Mdn = 55), Z= -0.586, p = .558, r = -0.113. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 17.5) and students who did not (Mdn = 14.5), Z= -1.885, p = .059, r = -0.37. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 19) and students who did not (Mdn = 18), Z= -0.648, p = .517, r = -0.13. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 16) and students who did not (Mdn = 16), Z= -0.074, p = .941, r = -0.014. These results are summarized in Table 4.9. Notice that N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.9.

Performance (Score) of Students Classified as Juniors on Standard Course Assessment Mechanisms

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	62	12	55	15	-0.586	0.558	-0.113
Programming Assignment 5	17.5	12	14.5	14	-1.885	0.059	-0.37
Programming Assignment 6	19	11	18	14	-0.648	0.517	-0.13
Programming Assignment 7	16	12	16	15	-0.074	0.941	-0.014

4.5.2.2.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did significantly differ between students who received the treatment (Mdn = 55 minutes) and students who did not (Mdn = 76 minutes), Z= -2.542, p = .011, r = -0.489. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did significantly differ between students who received the treatment (Mdn = 43 minutes) and students who did not (Mdn = 70 minutes), Z= -3.151, p = .002, r = -0.606. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task

did significantly differ between students who received the treatment (Mdn = 93 minutes) and students who did not (Mdn = 125 minutes), Z= - -2.84, p = .005, r = -0.733. Recall though that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Only 58% of treatment group attendees (7 students out of 12) completed the activity, while only 53% of control group attendees (8 students out of 15) completed it. The results of the Mann-Whitney tests are summarized in Table 4.10, while median time on task is illustrated in Figure 4.5.

Table 4.10.

Time on Task (Minutes) for Students Classified as Juniors Completing In-Laboratory Programming Activities

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	55	12	76	15	-2.542	0.011	-0.489
Programming Activity 6	43	12	70	15	-3.151	0.002	-0.606
Programming Activity 7	93	7	125	8	-2.84	0.005	-0.733

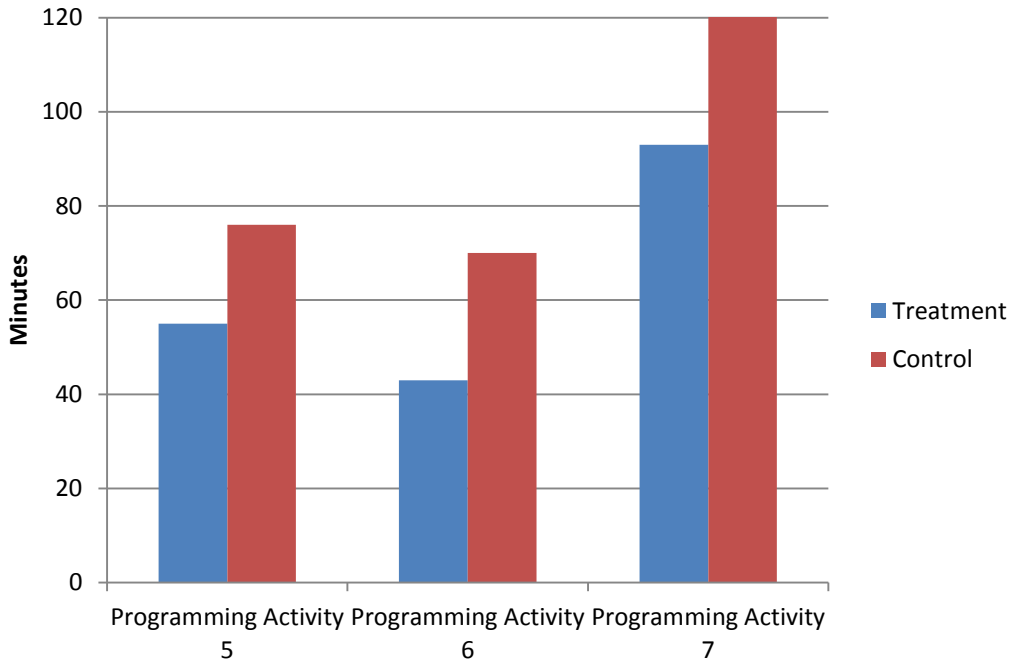


Figure 4.5. Median Time on Task for Students Classified as Juniors Completing In-Laboratory Programming Activities

4.5.2.2.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.17), Z= -0.605, p = .545, r = -0.121. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = .50) and students who did not (Mdn = 0.00), Z= -1.212, p = .226, r = -0.238. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0.13) and students who did not (Mdn = 0.00), Z= -0.434, p = .664, r = -0.095. The Mann-

Whitney test assessing change in the effect of group processing on the individual indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -0.682, p = .495, r = -0.131. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -1.114, p = .265, r = -0.214. The Mann-Whitney test assessing change in programming anxiety indicated that it did differ significantly between students who received the treatment (Mdn = 0.88) and students who did not (Mdn = 0.00), Z= -2.488, p = .013, r = -0.543. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -1.506, p = .132, r = -0.329. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.25), Z= -1.823, p = .068, r = -0.398. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -0.426, p = .67, r = -0.095. The Mann-Whitney test assessing change general self-efficacy indicated that it did differ significantly between students who received the treatment (Mdn = 0 .06) and students who did not (Mdn = -0.29), Z= -2.356, p = .018, r = -0.462. These results are summarized in Table 4.11.

Table 4.11.

Changes in the Self-Beliefs of Students Classified as Juniors during the Study

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	0.00	11	-0.17	14	-0.605	0.545	-0.121
Classroom Community	0.50	11	0.00	15	-1.212	0.226	-0.238
Debugging Self-Efficacy	0.13	10	0.00	11	-0.434	0.664	-0.095
Group Processing: Effect on Individual	0.00	12	0.00	15	-0.682	0.495	-0.131
Independence from/Dependence on Instructor	0.00	12	-0.33	15	-1.114	0.265	-0.214
Programming Anxiety	0.88	10	0.00	11	-2.488	0.013	-0.543
Programming Aptitude Mindset	0.00	10	-0.33	11	-1.506	0.132	-0.329
Programming Interest	0.00	10	-0.25	11	-1.823	0.068	-0.398
Programming Self-Concept	0.00	10	0.00	10	-0.426	0.67	-0.095
Self-Efficacy	0.06	12	-0.29	14	-2.356	0.018	-0.462

4.5.2.3 Students Classified as Sophomores

Twenty-five students in the course were classified as juniors. Eighteen were members of the treatment group, while seven were members of the control group.

4.5.2.3.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 41) and students who did not (Mdn = 67), Z= -0.908, p = .364, r = -0.182. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 16) and students who did not (Mdn = 17), Z= -0.249, p = .803, r = -0.053. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 17) and students who did not (Mdn = 19.5), Z= -1.764, p = .078, r = -0.394. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 14) and students who did not (Mdn = 19), Z= -1.793, p = .073, r = -0.366. These results are summarized in Table 4.12. As previously discussed, N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.12.

Performance (Score) of Students Classified as Sophomores on Standard Course Assessment Mechanisms

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	41	18	67	7	-0.908	0.364	-0.182
Programming Assignment 5	16	15	17	7	-0.249	0.803	-0.053
Programming Assignment 6	17	14	19.5	6	-1.764	0.078	-0.394
Programming Assignment 7	14	17	19	7	-1.793	0.073	-0.366

4.5.2.3.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 72 minutes) and students who did not (Mdn = 55 minutes), Z= -0.363, p = .716, r = -0.073. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 62 minutes) and students who did not (Mdn = 48 minutes), Z= -0.159, p = .874, r = -0.032. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task

did not significantly differ between students who received the treatment (Mdn =113 minutes) and students who did not (Mdn = 107 minutes), Z= -0.91, p = .363, r = -0.228. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Only 71% of treatment group attendees (12 students out of 17) completed the activity, and only 57% of control group attendees (4 students out of 7) completed it. The results of the Mann-Whitney tests are summarized in Table 4.13, while median time on task is illustrated in Figure 4.6.

Table 4.13.

Time on Task (Minutes) for Students Classified as Sophomores Completing In-Laboratory Programming Activities

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	72	18	55	7	-0.363	0.716	-0.073
Programming Activity 6	62	17	48	7	-0.159	0.874	-0.032
Programming Activity 7	113	12	107	4	-0.91	0.363	-0.228

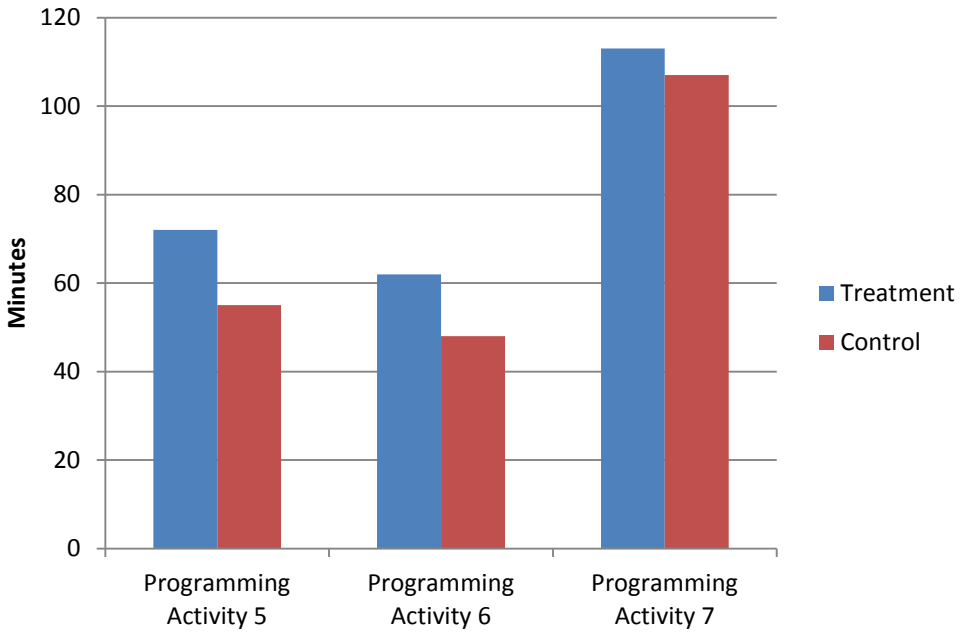


Figure 4.6. Median Time on Task for Students Classified as Sophomores Completing In-Laboratory Programming Activities

4.5.2.3.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = -0.17) and students who did not (Mdn = -0.33), Z= -0.781, p = .435, r = -0.166. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -0.577, p = .564, r = -0.118. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0.13) and students who did not (Mdn = 0.50), Z= -1.106, p = .269, r = -0.261.

The Mann-Whitney test assessing change in the effect of group processing on the individual indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -0.37, p = .711, r = -0.076. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -0.685, p = .493, r = -0.14. The Mann-Whitney test assessing change in programming anxiety indicated that it did differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.25), Z= -0.048, p = .962, r = -0.011. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = -0.33) and students who did not (Mdn = 0.00), Z= -1.114, p = .265, r = -0.255. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = -0.63) and students who did not (Mdn = 0.00), Z= -1.561, p = .119, r = -0.368. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = -0.25) and students who did not (Mdn = 0.00), Z= -0.177, p = .859, r = -0.041. The Mann-Whitney test assessing change general self-efficacy indicated that it did differ significantly between students who received the treatment (Mdn = -0.34) and students who did not (Mdn = 0.00), Z= -2.009, p = .045, r = -0.410. These results are summarized in Table 4.14.

Table 4.14.

Changes in the Self-Beliefs of Students Classified as Sophomores during the Study

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	-0.17	15	-0.33	7	-0.781	0.435	-0.166
Classroom Community	0.00	17	0.00	7	-0.577	0.564	-0.118
Debugging Self-Efficacy	0.13	12	0.50	6	-1.106	0.269	-0.261
Group Processing: Effect on Individual	0.00	17	0.00	7	-0.37	0.711	-0.076
Independence from/Dependence on Instructor	0.00	17	-0.33	7	-0.685	0.493	-0.14
Programming Anxiety	0.00	12	0.25	6	-0.048	0.962	-0.011
Programming Aptitude Mindset	-0.33	13	0.00	6	-1.114	0.265	-0.255
Programming Interest	-0.63	12	0.00	6	-1.561	0.119	-0.368
Programming Self-Concept	-0.25	13	0.00	6	-0.177	0.859	-0.041
Self-Efficacy	-0.34	17	0.00	7	-2.009	0.045	-0.41

4.5.3 Analysis of Treatment by Discipline

The data was analyzed to identify differential impact of the treatment on students enrolled in science disciplines and technology disciplines. Students in the science discipline had the following academic majors: Actuarial Science (17 students), and Mathematics (2 students). Students in the technology discipline had the following academic majors: Electrical Engineering Technology (1 student), Industrial Technology (4 students), Mechanical Engineering Technology (43 students), Manufacturing Engineering Technology (4 students), and Organizational Leadership (1 student). In addition, four non-science non-technology students were assigned to a category of 'Other'. The academic majors of these students were: Psychological Sciences (1 student), Pre-Communication (1 student), Exploratory Studies (1 student), and Marketing (1 student).

4.5.3.1 Students enrolled in Science Disciplines

Nineteen students in the course were enrolled in science disciplines. Fifteen were members of the treatment group, while four were members of the control group.

4.5.3.1.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 87) and students who did not (Mdn = 75.5), Z= -0.651, p = .515, r = -0.149. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who

received the treatment (Mdn = 19) and students who did not (Mdn = 18), Z= -0.921, p = .357, r = -0.211. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 20) and students who did not (Mdn = 18.5), Z= -0.973, p = .331, r = -0.229. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 18) and students who did not (Mdn = 17.5), Z= -0.46, p = .646, r = -0.106. These results are summarized in Table 4.15. As previously discussed, N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.15.

Performance (Score) of Students Enrolled in Science Disciplines on Standard Course Assessment Mechanisms

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	87	15	75.5	4	-0.651	0.515	-0.149
Programming Assignment 5	19	15	18	4	-0.921	0.357	-0.211
Programming Assignment 6	20	14	18.5	4	-0.973	0.331	-0.229
Programming Assignment 7	18	15	17.5	4	-0.46	0.646	-0.106

4.5.3.1.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 50 minutes) and students who did not (Mdn = 50 minutes), Z= -0.301, p = .763, r = -0.069. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 40 minutes) and students who did not (Mdn = 48 minutes), Z= -1.304, p = .192, r = -0.299. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task did not significantly differ between students who received the treatment (Mdn =101 minutes) and students who did not (Mdn = 113 minutes), Z= -0.813, p = .416, r = -0.226. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Only 73% of treatment group attendees (11 students out of 15) completed the activity, and only 50% of control group attendees (2 students out of 4) completed it. The results of the Mann-Whitney tests are summarized in Table 4.16, while median time on task is illustrated in Figure 4.7.

Table 4.16.

Time on Task (Minutes) for Students Enrolled in Science Disciplines Completing In-Laboratory Programming Activities

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	50	15	50	4	-0.301	0.763	-0.069
Programming Activity 6	40	15	48	4	-1.304	0.192	-0.299
Programming Activity 7	101	11	113	2	-0.813	0.416	-0.226

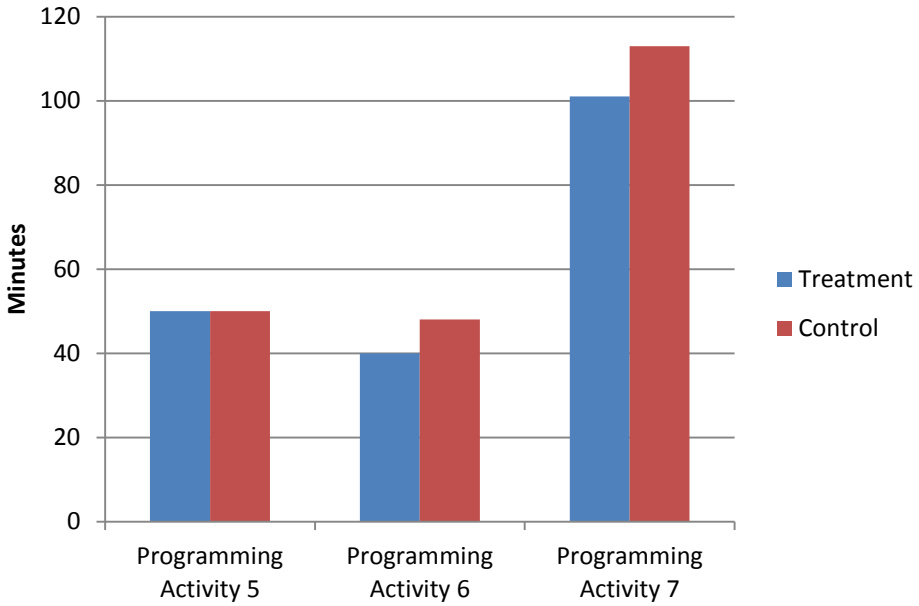


Figure 4.7. Median Time on Task for Students Enrolled in Science Disciplines Completing In-Laboratory Programming Activities.

4.5.3.1.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -0.34, p = .734, r = -0.085. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0. 25), Z= -0.298, p = .766, r = -0.07. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 25) and students who did not (Mdn = 0.75), Z= -0.711, p = .477, r = -0.19. The Mann-Whitney test assessing change in the effect of group processing on the individual indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 1.00), Z= -1.436, p = .151, r = -0.339. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did not differ significantly between students who received the treatment (Mdn = -0.33) and students who did not (Mdn = 0.00), Z= -0.607, p = .544, r = -0.143. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.50), Z= -0.274, p = .784, r = -0.068. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0. 67), Z= -1.315, p = .188, r = -0.329. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.25), Z= -0.205, p = .837, r = -0.051. The Mann-Whitney test assessing change in programming self-

concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0.25) and students who did not (Mdn = -0.25), Z= -1.871, p = .061, r = -0.468. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.11) and students who did not (Mdn = 0.00), Z= -0.254, p = .8, r = -0.062. These results are summarized in Table 4.17.

Table 4.17.

Changes in the Self-Beliefs of Students Enrolled in Science Disciplines during the Study

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	0	13	-0.33	3	-0.34	0.734	-0.085
Classroom Community	0	15	0.25	3	-0.298	0.766	-0.07
Debugging Self-Efficacy	0.25	11	0.75	3	-0.711	0.477	-0.19
Group Processing: Effect on Individual	0	15	1	3	-1.436	0.151	-0.339
Independence from/Dependence on Instructor	-0.33	15	0	3	-0.607	0.544	-0.143
Programming Anxiety	0	13	0.5	3	-0.274	0.784	-0.068
Programming Aptitude Mindset	0	13	0.67	3	-1.315	0.188	-0.329
Programming Interest	0	13	0.25	3	-0.205	0.837	-0.051
Programming Self-Concept	0.25	13	-0.25	3	-1.871	0.061	-0.468
Self-Efficacy	-0.11	14	0	3	-0.254	0.8	-0.062

4.5.3.2 Students enrolled in Technology Disciplines

Fifty-three students in the course were enrolled in technology disciplines. Twenty-nine were members of the treatment group, while twenty-four were members of the control group.

4.5.3.2.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 47) and students who did not (Mdn = 57), Z= -1.234, p = .217, r = -0.169. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 19) and students who did not (Mdn = 18), Z= -0.921, p = .357, r = -0.211. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did significantly differ between students who received the treatment (Mdn = 16) and students who did not (Mdn = 19), Z= -2.5, p = .012, r = - 0.365. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 15.5) and students who did not (Mdn = 16), Z= -1.543, p = .123, r = -0.216. These results are summarized in Table 4.18. As previously discussed, N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.18.

Performance (Score) of Students Enrolled in Technology Disciplines on Standard Course Assessment Mechanisms

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	47	29	57	24	-1.234	0.217	-0.169
Programming Assignment 5	16	26	15	22	-0.833	0.405	-0.12
Programming Assignment 6	16	25	19	22	-2.5	0.012	-0.365
Programming Assignment 7	15.5	28	16	23	-1.543	0.123	-0.216

4.5.3.2.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 61 minutes) and students who did not (Mdn = 75 minutes), Z= -1.788, p = .074, r = -0.246. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did significantly differ between students who received the treatment (Mdn = 57 minutes) and students who did not (Mdn = 69 minutes), Z= -2.539, p = .011, r = -0.355. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task

did significantly differ between students who received the treatment (Mdn =103 minutes) and students who did not (Mdn = 117 minutes), Z= -1.997, p = .046, r = -0.353. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Only 64% of treatment group attendees (18 students out of 28) completed the activity, and only 63% of control group attendees (14 students out of 22) completed it. The results of the Mann-Whitney tests are summarized in Table 4.19, while median time on task is illustrated in Figure 4.8.

Table 4.19.

Time on Task (Minutes) for Students Enrolled in Technology Disciplines Completing In-Laboratory Programming Activities

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	61	29	75	24	-1.788	0.074	-0.246
Programming Activity 6	57	28	69	23	-2.539	0.011	-0.355
Programming Activity 7	103	18	117	14	-1.997	0.046	-0.353

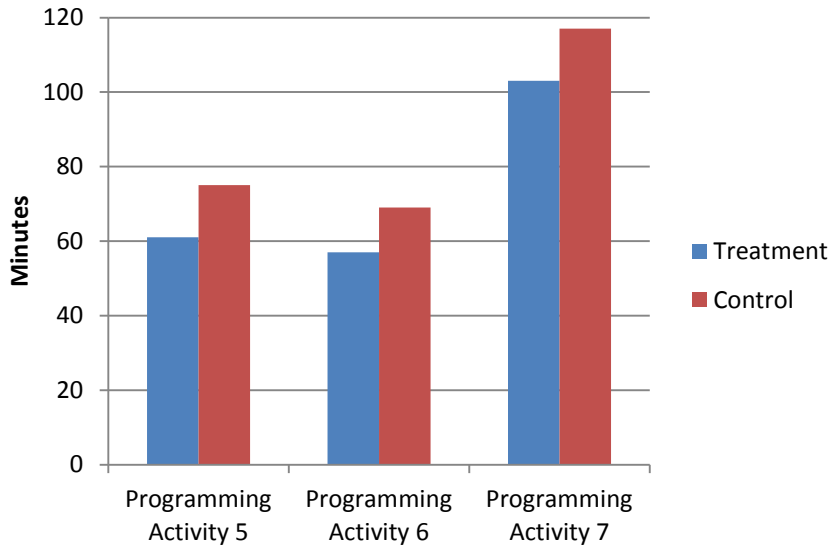


Figure 4.8. Median Time on Task for Students Enrolled in Technology Disciplines Completing In-Laboratory Programming Activities

4.5.3.2.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.25), Z= -1.717, p = .086, r = -0.248. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 13) and students who did not (Mdn = 0. 00), Z= -1.007, p = .314, r = -0.145. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 00) and students who did not (Mdn = 0. 25), Z= -1.443, p = .149, r = -0.255. The Mann-Whitney test assessing change in the effect of group processing on the individual indicated

that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -0.422, p = .673, r = -0.06. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -2.648, p = .008, r = -0.375. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.50), Z= -0.274, p = .784, r = -0.068. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0. 33), Z= -0.745, p = .457, r = -0.13. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = -0.25) and students who did not (Mdn = -0.25), Z= -0.655, p = .512, r = -0.116. The Mann-Whitney test assessing change in programming self-concept indicated that it did differ significantly between students who received the treatment (Mdn = -0.25) and students who did not (Mdn = 0.00), Z= -2.102, p = .036, r = -0.372. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.11) and students who did not (Mdn = -0.11), Z= 0.00, p = 1.00, r = 0.00. These results are summarized in Table 4.20.

Table 4.20.

Changes in the Self-Beliefs of Students Enrolled in Technology Disciplines during the Study

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	0.00	26	-0.25	22	-1.717	0.086	-0.248
Classroom Community	0.13	26	0.00	22	-1.007	0.314	-0.145
Debugging Self-Efficacy	0.00	19	0.25	13	-1.443	0.149	-0.255
Group Processing: Effect on Individual	0.00	27	0.00	22	-0.422	0.673	-0.06
Independence from/Dependence on Instructor	0.00	28	-0.33	22	-2.648	0.008	-0.375
Programming Anxiety	0.25	18	0.00	14	-1.3	0.194	-0.23
Programming Aptitude Mindset	0.00	19	-0.33	14	-0.745	0.457	-0.13
Programming Interest	-0.25	18	-0.25	14	-0.655	0.512	-0.116
Programming Self-Concept	-0.25	19	0.00	13	-2.102	0.036	-0.372
Self-Efficacy	-0.11	27	-0.11	21	0	1	0

4.5.4 Analysis of Treatment by Gender

The data was analyzed to identify differential impact of the treatment on female and male students.

4.5.4.1 Analysis of Treatment on Female Students

Nine students in the course were female. Six were members of the treatment group, while three were members of the control group.

4.5.4.1.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 71.5) and students who did not (Mdn = 76), Z= -0.516, p = .606, r = -0.172. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 19.5) and students who did not (Mdn = 16), Z= -1.313, p = .189, r = -0.438. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did significantly differ between students who received the treatment (Mdn = 20) and students who did not (Mdn = 18), Z= -0.853, p = .394, r = -0.284. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 18) and students who did not (Mdn = 19), Z= -0.795, p = .427, r = -0.265. These results are summarized in Table 4.21. As previously discussed, N varies in the data.

This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.21.

Performance (Score) of Female Students on Standard Course Assessment Mechanisms

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	71.5	6	76	3	-0.516	0.606	-0.172
Programming Assignment 5	19.5	6	16	3	-1.313	0.189	-0.438
Programming Assignment 6	20	6	18	3	-0.853	0.394	-0.284
Programming Assignment 7	18	6	19	3	-0.795	0.427	-0.265

4.5.4.1.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 54 minutes) and students who did not (Mdn = 82 minutes), Z= -1.296, p = .195, r = -0.432. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did significantly

differ between students who received the treatment (Mdn = 32 minutes) and students who did not (Mdn = 65 minutes), Z= -2.074, p = .038, r = -0.691. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task did not significantly differ between students who received the treatment (Mdn =97 minutes) and students who did not (Mdn = 148 minutes), Z= -1.514, p = .130, r = -0.572. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. One hundred percent of treatment group attendees (6 students out of 6) completed the activity, while only thirty-three percent of control group attendees (1 student out of 3) completed it. The results of the Mann-Whitney tests are summarized in Table 4.22, while median time on task is illustrated in Figure 4.9.

Table 4.22.

Time on Task (Minutes) for Female Students Completing In-Laboratory Programming Activities

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	54	6	82	3	-1.296	0.195	-0.432
Programming Activity 6	32	6	65	3	-2.074	0.038	-0.691
Programming Activity 7	97	6	148	1	-1.514	0.13	-0.572

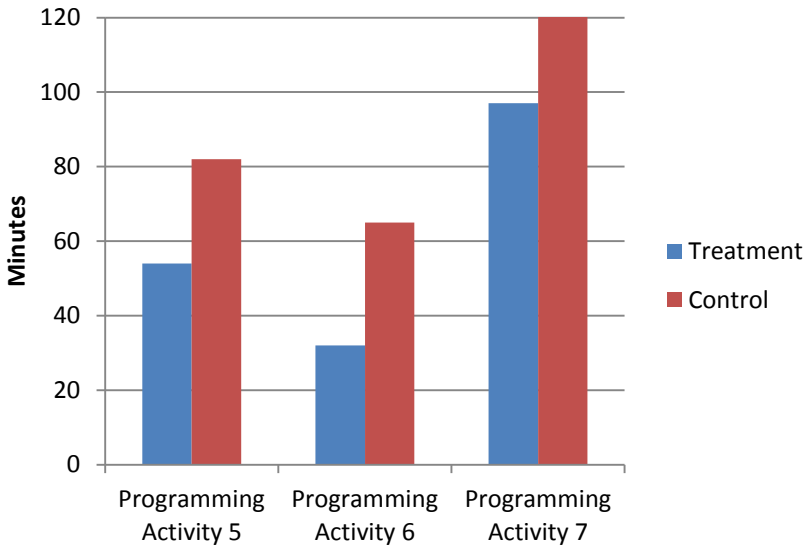


Figure 4.9. Median Time on Task for Female Students Completing In-Laboratory Programming Activities

4.5.4.1.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = -0.25) and students who did not (Mdn = -0.17), Z= 0.000, p = 1.000, r = 0.000. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 25) and students who did not (Mdn = 0. 00), Z= -0.263, p = .793, r = -0.088. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 25) and students who did not (Mdn = 0. 00), Z= -0.741, p = .459, r = -0.28. The Mann-Whitney test assessing change in the effect of group processing on the individual indicated that it

did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 1.00), Z= -0.894, p = .371, r = -0.298. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did not differ significantly between students who received the treatment (Mdn = -0.34) and students who did not (Mdn = 0.00), Z= -1.83, p = .067, r = -0.61. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 38) and students who did not (Mdn = 0.00), Z= -0.539, p = .59, r = -0.18. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0. 67), Z= -0.948, p = .343, r = -0.316. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.50), Z= -0.655, p = .512, r = -0.116. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0.25) and students who did not (Mdn = 0.00), Z= -1.307, p = .191, r = -0.436. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.11) and students who did not (Mdn = -0.56), Z= -1.042, p = .298, r = -0.347. These results are summarized in Table 4.23.

Table 4.23.

Changes in the Self-Beliefs of Female Students during the Study

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	-0.25	6	-0.17	3	0.000	1.000	0.000
Classroom Community	0.25	6	0.00	3	-0.263	0.793	-0.088
Debugging Self-Efficacy	0.25	4	0.00	3	-0.741	0.459	-0.28
Group Processing: Effect on Individual	0.00	6	1.00	3	-0.894	0.371	-0.298
Independence from/Dependence on Instructor	-0.34	6	0.00	3	-1.83	0.067	-0.61
Programming Anxiety	0.38	6	0.00	3	-0.539	0.59	-0.18
Programming Aptitude Mindset	0.00	6	0.67	3	-0.948	0.343	-0.316
Programming Interest	0.00	6	0.50	3	-1.625	0.104	-0.542
Programming Self-Concept	0.25	6	0.00	3	-1.307	0.191	-0.436
Self-Efficacy	-0.11	6	-0.56	3	-1.042	0.298	-0.347

4.5.4.2 Analysis of Treatment on Male Students

Sixty-seven students in the course were female. Thirty-nine were members of the treatment group, while twenty-eight were members of the control group.

4.5.4.2.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 60) and students who did not (Mdn = 57.5), Z= -0.28, p = .780, r = -0.034. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 17) and students who did not (Mdn = 15.5), Z= -1.491, p = .136, r = -0.189. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did significantly differ between students who received the treatment (Mdn = 17.5) and students who did not (Mdn = 19), Z= -1.718, p = .086, r = -0.222. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 16) and students who did not (Mdn = 16), Z= -1.046, p = .295, r = -0.130. These results are summarized in Table 4.24. As previously discussed, N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.24.

Performance (Score) of Male Students on Standard Course Assessment Mechanisms

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	60	39	57.5	28	-0.28	0.78	-0.034
Programming Assignment 5	17	36	15.5	26	-1.491	0.136	-0.189
Programming Assignment 6	17.5	34	19	26	-1.718	0.086	-0.222
Programming Assignment 7	16	38	16	27	-1.046	0.295	-0.13

4.5.4.2.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 59 minutes) and students who not (Mdn = 68 minutes), Z= -2.035, p = .042, r = -0.249. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did significantly differ between students who received the treatment (Mdn = 53 minutes) and students who did not (Mdn = 63 minutes), Z= -2.831, p = .005, r = -0.351. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task did not significantly differ between students who received the treatment (Mdn =105 minutes) and

students who did not (Mdn = 115 minutes), Z= -1.937, p = .053, r = -0.31. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Only 63% of treatment group attendees (24 students out of 38) completed the activity, while only 58% of control group attendees (15 students out of 26) completed it. The results of the Mann-Whitney tests are summarized in Table 4.25, while median time on task is illustrated in Figure 4.10.

Table 4.25.

Time on Task (Minutes) for Male Students Completing In-Laboratory Programming Activities

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	59	39	68	28	-2.035	0.042	-0.249
Programming Activity 6	53	38	63	27	-2.831	0.005	-0.351
Programming Activity 7	105	24	115	15	-1.937	0.053	-0.31

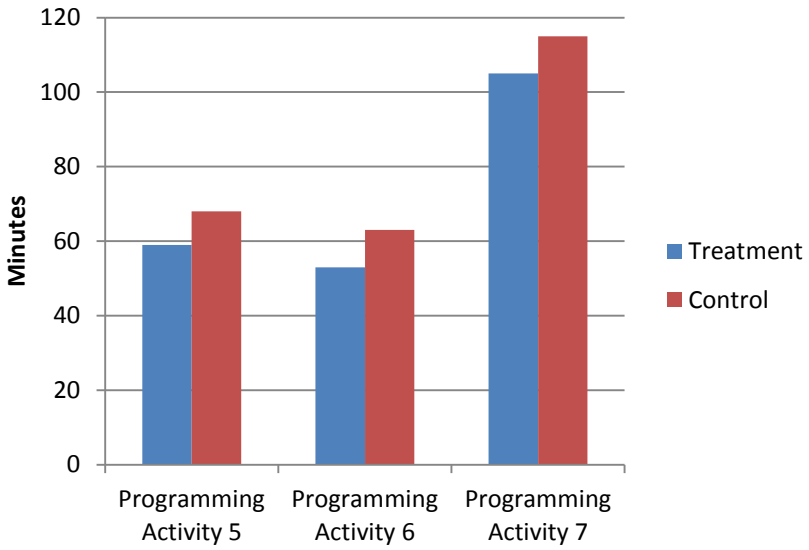


Figure 4.10. Median Time on Task for Male Students Completing In-Laboratory Programming Activities

4.5.4.2.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.25), Z= -1.75, p = .08, r = -0.23. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 00) and students who did not (Mdn = 0. 00), Z= -0.601, p = .548, r = -0.077. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 00) and students who did not (Mdn = 0. 25), Z= -1.809, p = .07, r = -0.276.

The Mann-Whitney test assessing change in the effect of group processing on the individual indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -0.477, p = .633, r = -0.061. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = - 0.33), Z= -2.912, p = .004, r = -0.367. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 13) and students who did not (Mdn = 0.00), Z= -1.201, p = .23, r = -0.183. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -0.638, p = .523, r = -0.096. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = -0.25) and students who did not (Mdn = -0.25), Z= 0.000, p = 1.000, r = 0.000. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -0.47, p = .638, r = -0.072. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.11) and students who did not (Mdn = -0.11), Z= -0.197, p = .844, r = -0.025. These results are summarized in Table 4.26.

Table 4.26.

Changes in the Self-Beliefs of Male Students during the Study

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	0.00	34	-0.25	24	-1.75	0.08	-0.23
Classroom Community	0.00	36	0.00	25	-0.601	0.548	-0.077
Debugging Self-Efficacy	0.00	27	0.25	16	-1.809	0.07	-0.276
Group Processing: Effect on Individual	0.00	37	0.00	25	-0.477	0.633	-0.061
Independence from/Dependence on Instructor	0.00	38	-0.33	25	-2.912	0.004	-0.367
Programming Anxiety	0.13	26	0.00	17	-1.201	0.23	-0.183
Programming Aptitude Mindset	0.00	27	-0.33	17	-0.638	0.523	-0.096
Programming Interest	-0.25	26	-0.25	17	0	1	0
Programming Self-Concept	0.00	27	0.00	16	-0.47	0.638	-0.072
Self-Efficacy	-0.11	36	-0.11	24	-0.197	0.844	-0.025

4.5.5 Analysis of Treatment by Performance on Prior Course Programming Examination

The data was analyzed to identify differential impact of the treatment on students demonstrating high programming performing level, moderate programming performance level, and low programming performance level on programming examination 1. Recall, programming examination 1 was administered immediately prior to the start of treatment in the study. High performers are those who scored 80 points or greater (corresponding to 80 % or greater) on programming examination 1. Moderate performers are those who scored between 50 points and 79 points (corresponding to 50%-79%) on programming examination 1. Low performers are those who scored below 60 points (less than 60%) on programming examination 1.

4.5.5.1 Students with High Performance on the Prior Programming Exam

Forty-six students in the course were considered to have high performance on programming examination 1, administered immediately prior to the start of the treatment. These forty-six students scored 80 points or greater (corresponding to 80 % or greater) on programming examination 1. Twenty-seven were assigned to of the treatment group, while nineteen were assigned to the control group.

4.5.5.1.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 78) and students who did not (Mdn = 70), $Z = -1.92$, $p = .055$, $r = -0.283$. The Mann-Whitney test assessing differences in student performance on programming assignment 5

indicated that performance did significantly differ between students who received the treatment (Mdn = 18.5) and students who did not (Mdn = 16), Z= -2.848, p = .004, r = -0.429. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did significantly differ between students who received the treatment (Mdn = 19) and students who did not (Mdn = 19), Z= -0.403, p = .687, r = -0.061. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 17) and students who did not (Mdn = 16.5), Z= -0.236, p = .814, r = -0.035. These results are summarized in Table 4.27. As previously discussed, N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.27.

Performance (Score) on Standard Course Assessment Mechanisms by Students with High Prior Programming Examination Performance

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	78	27	70	19	-1.92	0.055	-0.283
Programming Assignment 5	18.5	26	16	18	-2.848	0.004	-0.429
Programming Assignment 6	19	25	19	19	-0.403	0.687	-0.061
Programming Assignment 7	17	27	16.5	18	-0.236	0.814	-0.035

4.5.5.1.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 56 minutes) and students who did not (Mdn = 62 minutes), Z= -1.875, p = .061, r = -0.276. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did significantly differ between students who received the treatment (Mdn = 43 minutes) and students who did not (Mdn = 58 minutes), Z= -2.867, p = .004, r = -0.432. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task did significantly differ between students who received the treatment (Mdn =101 minutes) and students who did not (Mdn = 116 minutes), Z= -2.236, p = .025, r = -0.402. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Seventy percent of treatment group attendees (19 students out of 27) completed the activity, while only sixty-seven percent of control group attendees (12 students out of 18) completed it. The results of the Mann-Whitney tests are summarized in Table 4.28, while median time on task is illustrated in Figure 4.11.

Table 4.28.

Time on Task (Minutes) while Completing In-Laboratory Programming Activities for Students with High Prior Programming Examination Performance

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	56	27	62	19	-1.875	0.061	-0.276
Programming Activity 6	43	26	58	18	-2.867	0.004	-0.432
Programming Activity 7	101	19	116	12	-2.236	0.025	-0.402

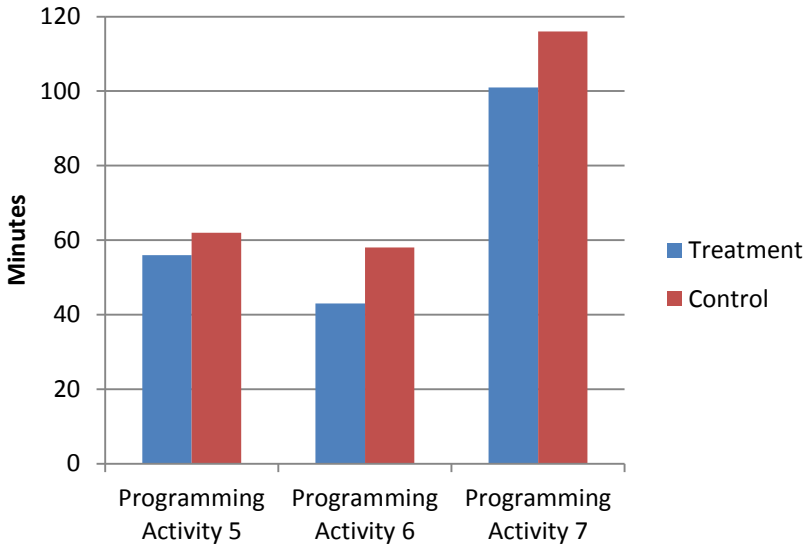


Figure 4.11. Median Time on Task while Completing In-Laboratory Programming Activities for Students with High Prior Programming Examination Performance.

4.5.5.1.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.17), Z= -1.419, p = .156, r = -0.227. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 38) and students who did not (Mdn = 0. 00), Z= -0.896, p = .370, r = -0.138. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 13) and students who did not (Mdn = 0. 25), Z= -0.52, p = .603, r = -0.088. The Mann-Whitney test assessing change in the effect of group processing on the individual indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -0.058, p = .954, r = -0.009. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -1.58, p = .114, r = -0.241. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 00) and students who did not (Mdn = 0.00), Z= -1.119, p = .263, r = -0.189. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -0.271, p = .786, r = -0.045. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -0.331, p = .741, r = -0.056. The Mann-Whitney test assessing change in programming self-

concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0.25) and students who did not (Mdn = 0.00), Z= -0.86, p = .39, r = -0.145. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.11) and students who did not (Mdn = 0.00), Z= -0.404, p = .686, r = -0.062. These results are summarized in Table 4.29.

Table 4.29.

Changes in the Self-Beliefs of Students with High Prior Programming Examination Performance.

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	0.00	24	-0.17	15	-1.419	0.156	-0.227
Classroom Community	0.38	26	0.00	16	-0.896	0.370	-0.138
Debugging Self-Efficacy	0.13	22	0.25	13	-0.52	0.603	-0.088
Group Processing: Effect on Individual	0.00	27	0.00	16	-0.058	0.954	-0.009
Independence from/Dependence on Instructor	0.00	27	-0.33	16	-1.58	0.114	-0.241
Programming Anxiety	0.00	22	0.00	13	-1.119	0.263	-0.189
Programming Aptitude Mindset	0.00	23	-0.33	13	-0.271	0.786	-0.045
Programming Interest	0.00	22	0.00	13	-0.331	0.741	-0.056
Programming Self-Concept	0.25	23	0.00	12	-0.86	0.390	-0.145
Self-Efficacy	-0.11	26	0.00	16	-0.404	0.686	-0.062

4.5.5.2 Students with Moderate Performance on the Prior Programming Exam

Twenty-one students in the course were considered to have moderate performance on programming examination 1, administered immediately prior to the start of the treatment. These students scored between 50 points and 79 points (corresponding to 50%-79%) on programming examination 1. Fourteen were assigned to of the treatment group, while seven were assigned to the control group

4.5.5.2.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 35) and students who did not (Mdn = 48), Z= -1.755, p = .079, r = -0.383. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did significantly differ between students who received the treatment (Mdn = 12) and students who did not (Mdn = 13), Z= -0.826, p = .409, r = -0.18. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did significantly differ between students who received the treatment (Mdn = 16) and students who did not (Mdn = 18), Z= -0.599, p = .549, r = -0.134. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 14) and students who did not (Mdn = 18), Z= -2.153, p = .031, r = -0.481. These results are summarized in Table 4.30. As previously discussed, N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.30.

Performance (Score) on Standard Course Assessment Mechanisms by Students with Moderate Prior Programming Examination Performance

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	35	14	48	7	-1.755	0.079	-0.383
Programming Assignment 5	12	14	13	7	-0.826	0.409	-0.18
Programming Assignment 6	16	13	18	7	-0.599	0.549	-0.134
Programming Assignment 7	14	13	18	7	-2.153	0.031	-0.481

4.5.5.2.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 62 minutes) and students who did not (Mdn = 76 minutes), Z= -0.523, p = .601, r = -0.114. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 62 minutes) and students who did not (Mdn = 70 minutes), Z= -1.46, p = .144, r = -0.319. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task

did not significantly differ between students who received the treatment (Mdn =115 minutes) and students who did not (Mdn = 109 minutes), Z= -0.472, p = .637, r = -0.142. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Sixty-four percent of treatment group attendees (9 students out of 14) completed the activity, while only thirty-three percent of control group attendees (2 students out of 6) completed it. The results of the Mann-Whitney tests are summarized in Table 4.31, while median time on task is illustrated in Figure 4.12.

Table 4.31.

Time on Task (Minutes) while Completing In-Laboratory Programming Activities for Students with Moderate Prior Programming Examination Performance

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	62	14	76	7	-0.523	0.601	-0.114
Programming Activity 6	62	14	70	7	-1.46	0.144	-0.319
Programming Activity 7	115	9	109	2	-0.472	0.637	-0.142

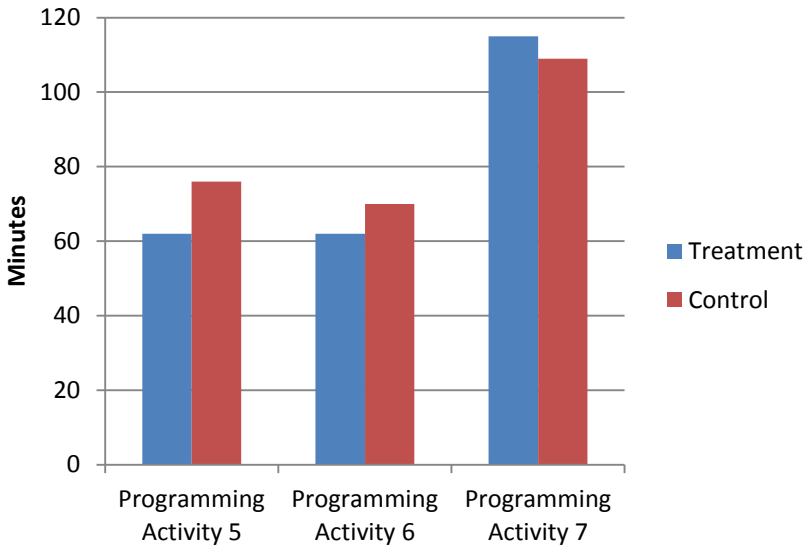


Figure 4.12. Median Time on Task while Completing In-Laboratory Programming Activities for Students with Moderate Prior Programming Examination Performance.

4.5.5.2.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = -0.16) and students who did not (Mdn = -0.33), Z= -1.352, p = .176, r = -0.302. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 00) and students who did not (Mdn = 0.25), Z= -1.084, p = .278, r = -0.242. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.13) and students who did not (Mdn = 0. 25), Z= -1.656, p = .098, r = -0.499. The Mann-Whitney test assessing change in the effect of group processing on the individual indicated that it did not differ significantly between students who received the treatment (Mdn =

0.00) and students who did not (Mdn = 0.00), Z= -0.567, p = .571, r = -0.13. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did not differ significantly between students who received the treatment (Mdn = 0.33) and students who did not (Mdn = -0.33), Z= -1.637, p = .102, r = -0.366. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 25) and students who did not (Mdn = 0.25), Z= -0.234, p = .815, r = -0.065. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.67), Z= -1.804, p = .071, r = -0.500. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = -0.25) and students who did not (Mdn = -0.50), Z= -0.237, p = .812, r = -0.066. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0.50) and students who did not (Mdn = 0.00), Z= -1.655, p = .098, r = -0.459. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.17) and students who did not (Mdn = -0.23), Z= -0.596, p = .551, r = 0.137. These results are summarized in Table 4.32.

Table 4.32.

Changes in the Self-Beliefs of Students with Moderate Prior Programming Examination Performance.

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	-0.16	13	-0.33	7	-1.352	0.176	-0.302
Classroom Community	0	13	0.25	7	-1.084	0.278	-0.242
Debugging Self-Efficacy	-0.13	8	0.25	3	-1.656	0.098	-0.499
Group Processing: Effect on Individual	0	12	0	7	-0.567	0.571	-0.13
Independence from/Dependence on Instructor	0.33	13	-0.33	7	-1.637	0.102	-0.366
Programming Anxiety	0.25	9	0.25	4	-0.234	0.815	-0.065
Programming Aptitude Mindset	0	9	-0.67	4	-1.804	0.071	-0.5
Programming Interest	-0.25	9	-0.5	4	-0.237	0.812	-0.066
Programming Self-Concept	-0.5	9	0	4	-1.655	0.098	-0.459
Self-Efficacy	-0.17	12	-0.23	7	-0.596	0.551	-0.137

4.5.5.3 Students with Low Performance on the Prior Programming Exam

Nine students in the course were considered to have Low performance on programming examination 1, administered immediately prior to the start of the treatment. These students scored less than 50 points (below 50%) on programming examination 1. Four were assigned to of the treatment group, while five were assigned to the control group

4.5.5.3.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 23) and students who did not (Mdn = 34), Z= -0.861, p = .389, r = -0.287. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 11.5) and students who did not (Mdn = 6.5), Z= -0.715, p = .475, r = -0.292. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 7.5) and students who did not (Mdn = 19), Z= -1.732, p = .083, r = -0.775. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 5.5) and students who did not (Mdn = 10), Z= -1.107, p = .268, r = -0.369. These results are summarized in Table 4.33. As previously discussed, N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.33.

Performance (Score) on Standard Course Assessment Mechanisms by Students with Low Prior Programming Examination Performance

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	23	4	34	5	-0.861	0.389	-0.287
Programming Assignment 5	11.5	2	6.5	4	-0.715	0.475	-0.292
Programming Assignment 6	7.5	2	19	3	-1.732	0.083	-0.775
Programming Assignment 7	5.5	4	10	5	-1.107	0.268	-0.369

4.5.5.3.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did significantly differ between students who received the treatment (Mdn = 68 minutes) and students who did not (Mdn = 87 minutes), Z= -2.46, p = .014, r = -0.82. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 47 minutes) and students who did not (Mdn = 75 minutes), Z= -1.715, p = .086, r = -0.572. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task

did not significantly differ between students who received the treatment (Mdn =102 minutes) and students who did not (Mdn = 129 minutes), Z= -1.549, p = .121, r = -0.775. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Sixty-seven percent of treatment group attendees (2 students out of 3) completed the activity, while only forty percent of control group attendees (2 students out of 5) completed it. The results of the Mann-Whitney tests are summarized in Table 4.34, while median time on task is illustrated in Figure 4.13.

Table 4.34.

Time on Task (Minutes) while Completing In-Laboratory Programming Activities for Students with Low Prior Programming Examination Performance

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	68	4	87	5	-2.46	0.014	-0.82
Programming Activity 6	47	4	75	5	-1.715	0.086	-0.572
Programming Activity 7	102	2	129	2	-1.549	0.121	-0.775

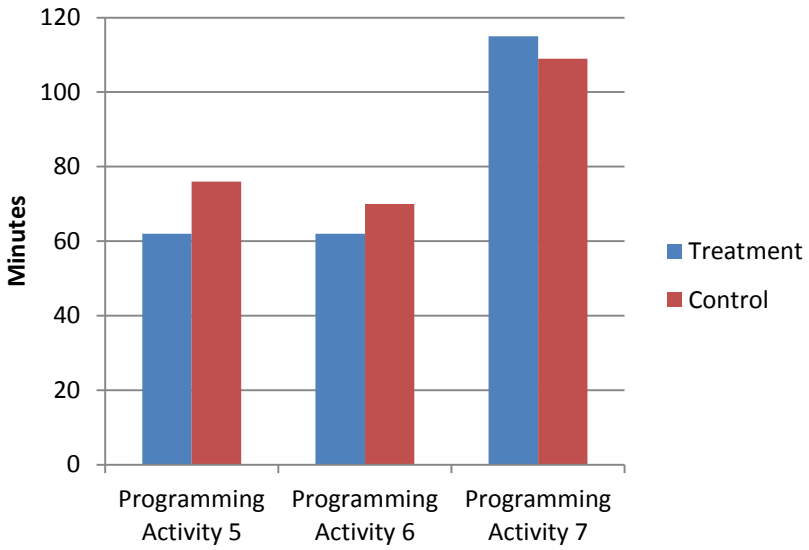


Figure 4.13. Median Time on Task while Completing In-Laboratory Programming Activities for Students with Low Prior Programming Examination Performance.

4.5.5.3.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = -0.33) and students who did not (Mdn = -0.17), Z= -1.056, p = .291, r = -0.373. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 25) and students who did not (Mdn = -0.25), Z= -1.207, p = .227, r = -0.427. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0.25) and students who did not (Mdn = 0. 25), Z= 0.000, p = 1.000, r = 0.000. The Mann-Whitney test assessing change in the effect of group processing on the individual

indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= 0.000, p = 1.000, r = 0.000. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did not differ significantly between students who received the treatment (Mdn = -0.33) and students who did not (Mdn = -0.33), Z= -0.822, p = .411, r = -0.274. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = 1. 25) and students who did not (Mdn = -0.50), Z= -1.342, p = .180, r = -0.671. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 1.33) and students who did not (Mdn = 0. 34), Z= -1.342, p = .180, r = -0.671. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = 1.25) and students who did not (Mdn = -0.25), Z= -1.342, p = .180, r = -0.671. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = -0.25) and students who did not (Mdn = 0.00), Z= -0.471, p = .637, r = -0.236. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.23) and students who did not (Mdn = -0.51), Z= -0.146, p = .884, r = -0.052. These results are summarized in Table 4.35.

Table 4.35.

Changes in the Self-Beliefs of Students with Low Prior Programming Examination Performance.

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	-0.33	3	-0.17	5	-1.056	0.291	-0.373
Classroom Community	0.25	3	-0.25	5	-1.207	0.227	-0.427
Debugging Self-Efficacy	0.25	1	0.25	3	0.000	1.00	0.000
Group Processing: Effect on Individual	0.00	4	0.00	5	0.000	1.00	0.000
Independence from/Dependence on Instructor	-0.33	4	-0.33	5	-0.822	0.411	-0.274
Programming Anxiety	1.25	1	-0.50	3	-1.342	0.18	-0.671
Programming Aptitude Mindset	1.33	1	0.34	3	-1.342	0.18	-0.671
Programming Interest	1.25	1	-0.25	3	-1.342	0.18	-0.671
Programming Self-Concept	-0.25	1	0.00	3	-0.471	0.637	-0.236
Self-Efficacy	-0.23	4	-0.51	4	-0.146	0.884	-0.052

4.5.6 Analysis of Treatment by Prior Programming Experience

The data was analyzed to identify differential impact of the treatment on students by the amount of programming experience they possessed prior to the start of the course. Students with significant prior programming experience are those self-reporting having previously completed two or more formal college computer programming courses, or who do significant computer programming in the workplace. Students with moderate prior programming experience are those self-reporting having previously completed one formal college computer programming course. Students with slight prior programming experience are those self-reporting having previously performed minor programming and scripting in a non-computing college course. Students with no prior programming experience are those self-reporting having no prior programming experience. Three students self-reported having significant prior programming experience; nine students self-reported having moderate prior programming experience; eighteen students self-reported having slight prior programming experience; and thirty-eight students self-reported having no prior programming experience. In addition, eight students either failed to self-report, or chose not to disclose, the amount of prior programming experience they possessed. Because of this, these eight students are not included in the differential analysis by prior programming experience.

4.5.6.1 Students with Moderate or Significant Prior Programming Experience

Three students in the course reported having significant programming experience prior to the start of the course, while nine students reported having moderate programming experience prior to the start of the course. For purposes of statistical analysis, these two groups were combined into a single group containing all twelve students possessing moderate or better

programming experience prior to the start of the course. Two students with significant prior programming experience and one student with moderate prior programming experience were assigned to the treatment group. One student with significant prior programming experience and six students with moderate prior programming experience were assigned to the control group. Thus treatment group contained five students, and the control contained seven students.

4.5.6.1.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 78) and students who did not (Mdn = 74), Z= -0.406, p = .685, r = -0.117. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 19) and students who did not (Mdn = 15), Z= -2.134, p = .033, r = -0.643. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 20) and students who did not (Mdn = 19), Z= -0.087, p = .931, r = -0.025. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 19) and students who did not (Mdn = 16), Z= -1.342, p = .180, r = -0.387. These results are summarized in Table 4.36. As previously discussed, N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.36.

Performance (Score) on Standard Course Assessment Mechanisms by Students with Moderate or Significant Prior Programming Experience

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	78	5	74	7	-0.406	0.685	-0.117
Programming Assignment 5	19	5	15	6	-2.134	0.033	-0.643
Programming Assignment 6	20	5	19	7	-0.087	0.931	-0.025
Programming Assignment 7	19	5	16	7	-1.342	0.180	-0.387

4.5.6.1.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did significantly differ between students who received the treatment (Mdn = 38 minutes) and students who did not (Mdn = 61 minutes), Z= -2.196, p = .028, r = -0.634. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did significantly differ between students who received the treatment (Mdn = 33 minutes) and students who did not (Mdn = 57 minutes), Z= -2.847, p = .004, r = -0.822. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task

did significantly differ between students who received the treatment (Mdn =95 minutes) and students who did not (Mdn = 125 minutes), Z= -2.46, p = .014, r = -0.82. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Eighty percent of treatment group attendees (4 students out of 5) completed the activity, while only seventy-two percent of control group attendees (5 students out of 7) completed it. The results of the Mann-Whitney tests are summarized in Table 4.37, while median time on task is illustrated in Figure 4.14.

Table 4.37.

Time on Task (Minutes) while Completing In-Laboratory Programming Activities for Students with Moderate or Significant Prior Programming Experience

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	38	5	61	7	-2.196	0.028	-0.634
Programming Activity 6	33	5	57	7	-2.847	0.004	-0.822
Programming Activity 7	95	4	125	5	-2.46	0.014	-0.82

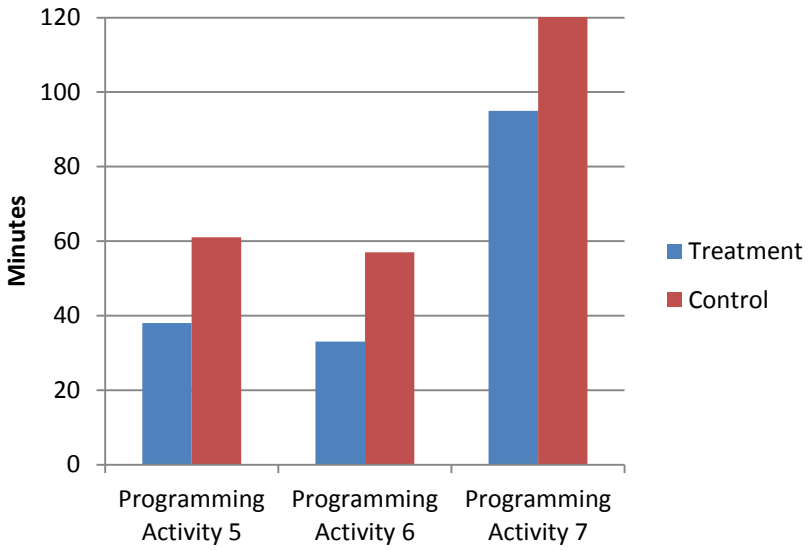


Figure 4.14. Median Time on Task while Completing In-Laboratory Programming Activities for Students with Moderate or Significant Prior Programming Experience.

4.5.6.1.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = -0.17) and students who did not (Mdn = -0.25), Z= -0.555, p = .579, r = -0.167. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 50) and students who did not (Mdn = -0.13), Z= -0.751, p = .453, r = -0.226. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0.25) and students who did not (Mdn = -0.13), Z= -0.754, p = .451, r = -0.251. The Mann-Whitney test assessing change in the effect of group processing on the individual indicated that it did not differ significantly between students who received the treatment (Mdn =

0.00) and students who did not (Mdn = 0.00), Z= -0.583, p = .560, r = -0.176. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -0.835, p = .404, r = -0.252. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 50) and students who did not (Mdn = 0.00), Z= -0.997, p = .319, r = -0.332. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = -0.33) and students who did not (Mdn = -0.33), Z= -0.135, p = .893, r = -0.045. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.38), Z= -0.876, p = .381, r = -0.292. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0.25) and students who did not (Mdn = 0.50), Z= -0.450, p = .653, r = -0.159. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.11), Z= -0.186, p = .852, r = -0.056. These results are summarized in Table 4.38.

Table 4.38.

Changes in the Self-Beliefs of Students with Moderate or Significant Prior Programming Experience.

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	-0.17	5	-0.25	6	-0.555	0.579	-0.167
Classroom Community	0.50	5	-0.13	6	-0.751	0.453	-0.226
Debugging Self-Efficacy	0.25	5	-0.13	4	-0.754	0.451	-0.251
Group Processing: Effect on Individual	0.00	5	0.00	6	-0.583	0.560	-0.176
Independence from/Dependence on Instructor	0.00	5	-0.33	6	-0.835	0.404	-0.252
Programming Anxiety	0.50	5	0.00	4	-0.997	0.319	-0.332
Programming Aptitude Mindset	-0.33	5	-0.33	4	-0.135	0.893	-0.045
Programming Interest	0.00	5	-0.38	4	-0.876	0.381	-0.292
Programming Self-Concept	0.25	5	0.50	3	-0.450	0.653	-0.159
Self-Efficacy	0.00	5	-0.11	6	-0.186	0.852	-0.056

4.5.6.2 Students with Slight Prior Programming Experience

Eighteen students self-reported having slight programming experience prior to the start of the course. Eleven were members of the treatment group, while seven were members of the control group.

4.5.6.2.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 74) and students who did not (Mdn = 67), Z= -0.86, p = .390, r = -0.203. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 18) and students who did not (Mdn = 16), Z= -1.528, p = .127, r = -0.371. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 19.5) and students who did not (Mdn = 19), Z= -0.626, p = .531, r = -0.156. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 17) and students who did not (Mdn = 16), Z= -0.663, p = .508, r = -0.161. These results are summarized in Table 4.39. As previously discussed, N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.39.

Performance (Score) on Standard Course Assessment Mechanisms by Students with Slight Prior Programming Experience

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	74	11	67	7	-0.86	0.39	-0.203
Programming Assignment 5	18	10	16	7	-1.528	0.127	-0.371
Programming Assignment 6	19.5	10	19	6	-0.626	0.531	-0.156
Programming Assignment 7	17	11	16	6	-0.663	0.508	-0.161

4.5.6.2.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did significantly differ between students who received the treatment (Mdn = 54 minutes) and students who did not (Mdn = 63 minutes), Z= -1.636, p = .102, r = -0.386. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 41 minutes) and students who did not (Mdn = 63 minutes), Z= -1.859, p = .063, r = -0.438. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task

did significantly differ between students who received the treatment (Mdn =87 minutes) and students who did not (Mdn = 110 minutes), Z= -1.793, p = .073, r = -0.497. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Seventy percent of treatment group attendees (7 students out of 10) completed the activity, while eighty-six percent of control group attendees (6 students out of 7) completed it. The results of the Mann-Whitney tests are summarized in Table 4.40, while median time on task is illustrated in Figure 4.15.

Table 4.40.

Time on Task (Minutes) while Completing In-Laboratory Programming Activities for Students with Slight Prior Programming Experience

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	54	11	63	7	-1.636	0.102	-0.386
Programming Activity 6	41	11	63	7	-1.859	0.063	-0.438
Programming Activity 7	87	7	110	6	-1.793	0.073	-0.497

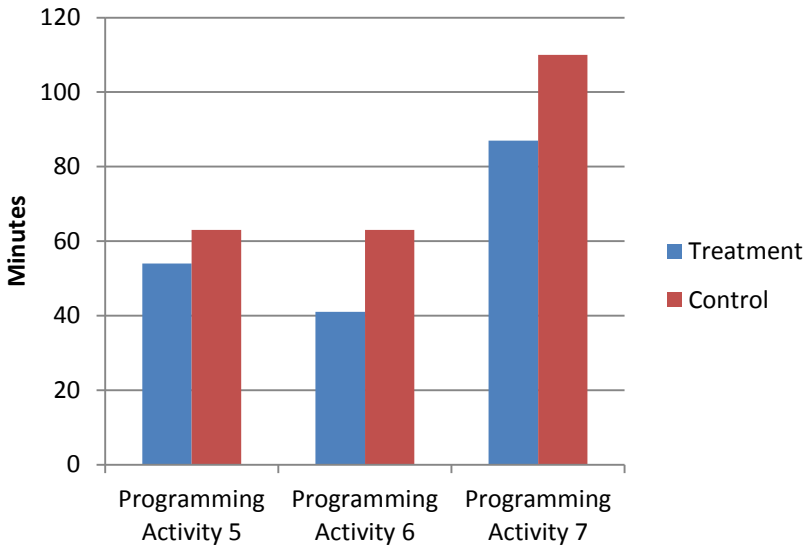


Figure 4.15. Median Time on Task while Completing In-Laboratory Programming Activities for Students with Slight Prior Programming Experience.

4.5.6.2.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = 0.08) and students who did not (Mdn = -0.33), Z= -1.477, p = .140, r = -0.358. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 00) and students who did not (Mdn = 0.00), Z= -0.369, p = .712, r = -0.087. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0.25) and students who did not (Mdn = 0.75), Z= -0.528, p = .598, r = -0.176. The Mann-Whitney test assessing change in the effect of group processing on the individual

indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -0.86, p = .390, r = -0.209. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -1.199, p = .230, r = -0.283. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 75) and students who did not (Mdn = 0.25), Z= -1.384, p = .166, r = -0.438. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.00), Z= -1.224, p = .221, r = -0.387. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = -0.13) and students who did not (Mdn = 0.25), Z= -0.802, p = .423, r = -0.267. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.25), Z= -1.302, p = .193, r = -0.412. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.11) and students who did not (Mdn = -0.11), Z= -0.753, p = .452, r = -0.188. These results are summarized in Table 4.41.

Table 4.41.

Changes in the Self-Beliefs of Students with Slight Prior Programming Experience.

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	0.08	10	-0.33	7	-1.477	0.14	-0.358
Classroom Community	0.00	11	0.00	7	-0.369	0.712	-0.087
Debugging Self-Efficacy	0.25	6	0.75	3	-0.528	0.598	-0.176
Group Processing: Effect on Individual	0.00	10	0.00	7	-0.860	0.390	-0.209
Independence from/Dependence on Instructor	0.00	11	-0.33	7	-1.199	0.230	-0.283
Programming Anxiety	0.75	7	0.25	3	-1.384	0.166	-0.438
Programming Aptitude Mindset	0.00	7	0.00	3	-1.224	0.221	-0.387
Programming Interest	-0.13	6	0.25	3	-0.802	0.423	-0.267
Programming Self-Concept	0.00	7	-0.25	3	-1.302	0.193	-0.412
Self-Efficacy	-0.11	9	-0.11	7	-0.753	0.452	-0.188

4.5.6.3 Students with No Prior Programming Experience

Thirty-eight students self-reported having no programming experience prior to the start of the course. Twenty-four were members of the treatment group, while fourteen were members of the control group.

4.5.6.3.1 *Course Performance Data*

A series of Mann-Whitney tests were employed to determine if performance (i.e., test score, homework score) differed between the treatment group and the control. The Mann-Whitney test assessing differences in student performance on programming examination 2 indicated that performance did not significantly differ between students who received the treatment (Mdn = 59) and students who did not (Mdn = 48.5), Z= -1.06, p = .289, r = -0.172. The Mann-Whitney test assessing differences in student performance on programming assignment 5 indicated that performance did not significantly differ between students who received the treatment (Mdn = 17) and students who did not (Mdn = 15), Z= -0.877, p = .380, r = -0.146. The Mann-Whitney test assessing differences in student performance on programming assignment 6 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 17) and students who did not (Mdn = 19), Z= -1.511, p = .131, r = -0.255. The Mann-Whitney test assessing differences in student performance on programming assignment 7 also indicated that performance did not significantly differ between students who received the treatment (Mdn = 16) and students who did not (Mdn = 18), Z= -1.628, p = .104, r = -0.268. These results are summarized in Table 4.42. As previously discussed, N varies in the data. This reflects students failing to submit homework assignments, as only those students who submit an assessment are counted in the sample for that assessment.

Table 4.42.

Performance (Score) on Standard Course Assessment Mechanisms by Students with No Prior Programming Experience

	Treatment		Control				
Performance Assessment	Mdn	N	Mdn	N	Z	p	r
Programming Exam 2	59	24	48.5	14	-1.06	0.289	-0.172
Programming Assignment 5	17	23	15	13	-0.877	0.380	-0.146
Programming Assignment 6	17	22	19	13	-1.511	0.131	-0.255
Programming Assignment 7	16	23	18	14	-1.628	0.104	-0.268

4.5.6.3.2 *Observational Data*

Another series of Mann-Whitney tests were employed to determine if time on task (i.e., number of minutes required to complete a programming activity) differed between the treatment group and the control. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 5 indicated that time on task did not significantly differ between students who received the treatment (Mdn = 63 minutes) and students who did not (Mdn = 81 minutes), Z= -1.847, p = .065, r = -0.300. The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 6 indicated that time on task did significantly differ between students who received the treatment (Mdn = 60 minutes) and students who did not (Mdn = 70 minutes), Z= -2.948, p = .003, r = -0.485. Finally, The Mann-Whitney test assessing differences in time on task for in-laboratory programming activity 7 indicated that time on task

did not significantly differ between students who received the treatment (Mdn =115 minutes) and students who did not (Mdn = 116 minutes), Z= -1.142, p = .253, r = -0.277. Recall that in-laboratory programming activity 7 was an intentionally long assignment meant to challenge student's abilities to create programs efficiently. Fifty-eight percent of treatment group attendees (14 students out of 24) completed the activity, while twenty-three percent of control group attendees (3 students out of 13) completed it. The results of the Mann-Whitney tests are summarized in Table 4.43, while median time on task is illustrated in Figure 4.16.

Table 4.43.

Time on Task (Minutes) while Completing In-Laboratory Programming Activities for Students with No Prior Programming Experience

	Treatment		Control				
Performance Activity	Mdn	N	Mdn	N	Z	p	r
Programming Activity 5	63	24	81	14	-1.847	0.065	-0.300
Programming Activity 6	60	23	70	14	-2.948	0.003	-0.485
Programming Activity 7	115	14	116	3	-1.142	0.253	-0.277

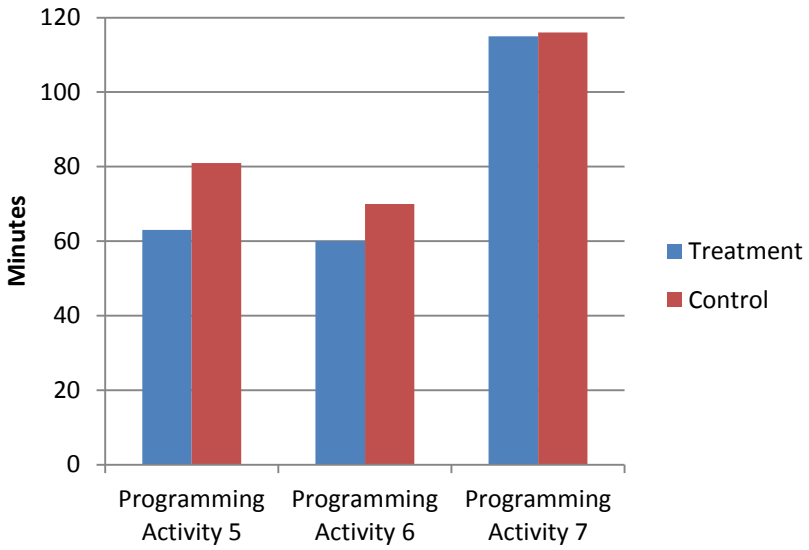


Figure 4.16. Median Time on Task while Completing In-Laboratory Programming Activities for Students with No Prior Programming Experience.

4.5.6.3.3 *Questionnaire Data*

A series of Mann-Whitney tests were employed to determine if the self-belief changes that students undergo during the learning process differed between the treatment group and the control group. The Mann-Whitney test assessing change in academic enjoyment indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.17), Z= -0.471, p = .638, r = -0.082. The Mann-Whitney test assessing change in feeling of classroom community indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 00) and students who did not (Mdn = 0.00), Z= -0.173, p = .863, r = -0.029. The Mann-Whitney test assessing change in debugging self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = 0.25), Z= -1.366, p = .172, r = -0.258. The Mann-Whitney test assessing change in the effect of group processing on the individual indicated that it did not differ significantly between students who received the treatment (Mdn =

0.00) and students who did not (Mdn = 0.00), Z= -0.366, p = .714, r = -0.060. The Mann-Whitney test assessing change in sense of dependence on the instructor for assistance indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.33), Z= -1.017, p = .309, r = -0.167. The Mann-Whitney test assessing change in programming anxiety indicated that it did not differ significantly between students who received the treatment (Mdn = 0. 00) and students who did not (Mdn =- 0.25), Z= -1.048, p = .295, r = -0.195. The Mann-Whitney test assessing change in programming aptitude mindset indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.34), Z= -0.873, p = .382, r = -0.159. The Mann-Whitney test assessing change in programming interest indicated that it did not differ significantly between students who received the treatment (Mdn = -0.25) and students who did not (Mdn =- 0.25), Z= -0.132, p = .895, r = -0.024. The Mann-Whitney test assessing change in programming self-concept indicated that it did not differ significantly between students who received the treatment (Mdn = 0.00) and students who did not (Mdn = -0.25), Z= -0.350, p = .726, r = -0.064. The Mann-Whitney test assessing change general self-efficacy indicated that it did not differ significantly between students who received the treatment (Mdn = -0.22) and students who did not (Mdn = -0.29), Z= -0.69, p = .490, r = -0.115. These results are summarized in Table 4.44.

Table 4.44.

Changes in the Self-Beliefs of Students with No Prior Programming Experience.

	Treatment		Control				
Self-Belief	Mdn Δ	N	Mdn Δ	N	Z	p	r
Academic Enjoyment	0.00	21	-0.17	12	-0.471	0.638	-0.082
Classroom Community	0.00	22	0.00	13	-0.173	0.863	-0.029
Debugging Self-Efficacy	0.00	18	0.25	10	-1.366	0.172	-0.258
Group Processing: Effect on Individual	0.00	24	0.00	13	-0.366	0.714	-0.060
Independence from/Dependence on Instructor	0.00	24	-0.33	13	-1.017	0.309	-0.167
Programming Anxiety	0.00	18	-0.25	11	-1.048	0.295	-0.195
Programming Aptitude Mindset	0.00	19	-0.34	11	-0.873	0.382	-0.159
Programming Interest	-0.25	19	-0.25	11	-0.132	0.895	-0.024
Programming Self-Concept	0.00	19	0.00	11	-0.35	0.726	-0.064
Self-Efficacy	-0.22	24	-0.29	12	-0.69	0.490	-0.115

4.6 Summary

This chapter has identified software used during data analysis of this study. This chapter subsequently discussed the treatment and control groups, participant demographics, and the data collection schedule. Finally, this chapter systematically approached and identified the results of the study, both overall and by differential impact.

CHAPTER 5: CONCLUSIONS, DISCUSSION, AND RECOMMENDATIONS

This chapter integrates and interprets the data presented in the prior chapter. The findings of this investigation, the impact of employing the instructional scaffold on the learning of procedural knowledge and programming-related self-beliefs, are first addressed. Discussion of these findings, and well as implications of these findings, follow. Lastly, recommendations for practitioners are provided and areas for future identified.

5.1 Conclusions

5.1.1 Research Questions 1 and 5

An overall view of the data collected during this case study reveals that pair programming, employed as an instructional scaffold, did impact the learning process of students. What is interesting is that this impact was not evident in the standard course performance data; students who received this additional instructional scaffolding during their laboratory periods displayed no significant difference from their non-scaffolded peers in course programming examination performance or course homework assignment performance. The impact of employing the pair programming methodology as an instructional scaffold only became evident through observationally collected data: time on task for in-laboratory performance tasks, and number of questions posed by students to instructors while completing in-laboratory performance tasks. The difference in time on task between those scaffolded and those not scaffolded was statistically significant. Those scaffolded benefiting from a decrease of 11 minutes in median time

on task for the in-laboratory 05 assignment, a performance task requiring 68 minutes of those unscaffolded; a decrease of 15 minutes in median time on task for the in-laboratory 06 assignment, a performance task requiring 64 minutes of those unscaffolded; and a decrease of 13 minutes in median time on task for the in-laboratory 07 assignment, a performance task requiring 116 minutes of those unscaffolded. This last performance task, in-laboratory assignment 07, was extremely lengthy: only 68% of those in the scaffolded group and 55% of those in the unscaffolded group finishing. These results indicate that, for a comparable amount of knowledge learned, learning occurred faster in those students who were instructionally scaffolded through use of pair programming. This implies that pair programming employed as an instructional scaffold increased the efficiency of learning of those students so scaffolded.

5.1.2 Research Questions 2 and 6

An overall view of the data collected during this case study reveals that pair programming employed as an instructional scaffold impacted the change of only one programming-related or course-related student self-belief: dependence on the instructor for assistance. Those students benefiting from the additional scaffolding of pair programming in their computer laboratories reported no median increase in their dependence on the instructor for assistance as the semester progressed. However, the students who did not receive the additional scaffolding of pair programming (i.e., they worked alone in their computer laboratories) did report a median increase in dependence on the instructor for assistance, the difference of which was statistically significant when compared with the scaffolded group. Observational data compliments these results, as students in the scaffolded group posed fewer questions to instructors while completing the in-laboratory performance tasks. Those scaffolded posed only 0.87 questions per student during completion of the in-laboratory 05 performance task while

those unscaffolded posed 1.34 questions per student; 0.82 questions per student during completion of the in-laboratory 06 performance task while those unscaffolded posed 2.55 questions per student; and 1.82 questions per student during completion of the in-laboratory 07 performance task while those unscaffolded posed 4.28 questions per student. These results imply that pair programming employed as an instructional scaffold decreased the dependency of students so scaffolded on the instructor during the learning process.

In contrast, no statistically significant difference in programming-related or course-related self-belief changes was evident from the data. This includes self-beliefs regarding academic enjoyment, classroom community, debugging self-efficacy, group/team work, anxiety caused by computer programming, mindset regarding computer programming aptitude, interest in computer programming, programming self-concept, and general academic self-efficacy.

5.1.3 Research Questions 3 and 4

5.1.3.1 Differential Impact on Students by Academic Classification

5.1.3.1.1 *Seniors*

Factor specific analysis of the data revealed that students classified as seniors who were scaffolded through employment of pair programming in the course laboratories benefitted overall by a statistically significant decrease in time on task when compared with other students, also classified as seniors, who did not pair program in the course laboratories. Students classified as seniors who were scaffolded through employment of pair programming in the course laboratories also experienced no change in sense of dependence on the instructor during study. This is in contrast to the increasing sense of dependence experienced by those students, also classified as seniors, who did not pair program in the course laboratories. However, no statistically significant

difference was observed in other self-beliefs, course examination performance or course homework performance. These senior-specific findings align with the overall course findings.

5.1.3.1.2 *Juniors*

Factor specific analysis of the data revealed that students classified as juniors who were scaffolded through employment of pair programming in the course laboratories benefitted overall by a statistically significant decrease in time on task when compared with other students, also classified as juniors, who did not pair program. Students classified as juniors that were scaffolded through pair programming also differed from their non-scaffolded peers in their changing sense of programming-induced anxiety and self-efficacy. In both cases, these differences were beneficial to those scaffolded. No statistically significant difference was observed in other self-beliefs, course examination performance or course homework performance, though changes in sense of dependence were nearly the level needed to rate statistical significance. Differential impact of the scaffold on sense of programming anxiety and sense of self-efficacy were thus observed in students classified as juniors.

5.1.3.1.3 *Sophomores*

Factor specific analysis of the data revealed that students classified as sophomores who were scaffolded through employment of pair programming in the course laboratories showed no statistically significant difference in time on task, course examination performance or course homework performance when compared with other students, also classified as sophomores, who did not pair program. Students classified as sophomores who were scaffolded did however experience a decreased sense of general self-efficacy when compared to their also sophomore peers. This difference was statistically significant, and detriment (not beneficial) to the students scaffolded. No other statistically significant differences were observed in self-belief changes.

Detrimental differential impact of the scaffold was thus observed on sense of self-efficacy in sophomores.

5.1.3.2 Differential Impact on Students by Academic Discipline

5.1.3.2.1 *Science Disciplines*

Factor specific analysis of the data revealed no statistically significant differences between those students enrolled in science disciplines that were scaffolded through employment of pair programming in the laboratory and those that were not. Differences in time on task were not found to be statistically significant. Differences in course examination performance and homework assignment performance were not found to be statistically significant. Differences in changes in self-beliefs were not found to be statistically significant. Differential impact was thus observed, with the programming methodology appearing to have no impact as a scaffold for science students.

5.1.3.2.2 *Technology Disciplines*

Factor specific analysis of the data reveals that students enrolled in technology disciplines that were scaffolded through employment of pair programming in the course laboratories benefitted overall by a statistically significant decrease in time on task when compared to technology students that did not pair program. Technology students who pair programmed also experienced no change in sense of dependence on the instructor during the study, in contrast to the increasing sense of dependence experienced by those technology students who did not pair program. However, they also experienced a decreasing sense of programming self-concept while their peers experienced no change in programming self-concept. No statistically significant difference was observed in other self-beliefs, course examination performance or course

homework performance. Detrimental differential impact of the scaffold was thus observed on sense of programming self-concept in technology students.

5.1.3.3 Differential Impact on Students by Gender

5.1.3.3.1 *Female Students*

Factor specific analysis of the data for female students was hampered by the small number of female students in the course (the control group had only 3 females). Though descriptive statistics imply impact of pair programming on female students align with the overall impact, comparative statistics revealed no statistically significant difference between those who pair programmed and those who did not.

5.1.3.3.2 *Male Students*

Factor specific analysis of the data revealed that male students scaffolded through employment of pair programming in the course laboratories benefitted overall by a statistically significant decrease in time on task when compared with their male peers that did not pair program. Male students that pair programmed also experienced no change in sense of dependence on the instructor during study. This is in contrast to the increasing sense of dependence experienced by those male students who did not pair program. No statistically significant difference was observed in other self-beliefs, course examination performance or course homework performance. These senior-specific findings align with the overall course findings.

5.1.3.4 Differential Impact on Students by Prior Exam Performance

5.1.3.4.1 *High Performance*

Factor specific analysis of the data revealed that students having high performance on programming examination 1, when scaffolded by employment of pair programming in the course laboratories, benefitted overall by a statistically significant decrease in time on task when compared with their high performing peers that did not pair program. No statistically significant difference was observed in other self-beliefs, course examination performance or course homework performance. Sense of dependence on the instructor approached, but failed to reach, statistical significance however. These findings align with the overall course findings.

5.1.3.4.2 *Moderate Performance*

Factor specific analysis of the data revealed no statistically significant differences between those students having moderate performance on programming examination 1 that were scaffolded through use of pair programming and those moderate performing students that did not. Differences in time on task were not found to be statistically significant. Differences in course examination performance and homework assignment performance, though they did approach statistical significance, were not found to be statistically significant. Differences in changes in self-beliefs were not found to be statistically significant. Differential impact was thus observed, with the programming methodology appearing to have no impact as a scaffold for these students.

5.1.3.4.3 *Low Performance*

Factor specific analysis of the data for students that had low performance on programming examination 1 was hampered by their initial small number, and compounded by the tendency of these students not to attend the course laboratory or turn in course assignments. Though descriptive statistics imply impact of pair programming on low performing students align

with the overall impact, comparative statistics revealed no statistically significant difference between those that pair programmed and those who did not.

5.1.3.5 Differential Impact on Students by Amount of Prior Programming Experience

5.1.3.5.1 *Moderate or More*

Factor specific analysis of the data revealed that students self-identifying as having a moderate or significant amount of programming experience prior to the start of the course, when scaffolded by employment of pair programming in the course laboratories, benefitted overall by a statistically significant decrease in time on task when compared with peers that did not pair program. No statistically significant difference was observed in other self-beliefs, course examination performance or course homework performance. These findings align with the overall course findings.

5.1.3.5.2 *Slight*

Factor specific analysis of the data revealed that students self-identifying as having a slight amount of programming experience prior to the start of the course, when scaffolded by employment of pair programming in the course laboratories, benefitted overall by a decrease in time on task when compared with peers that did not pair program. This decrease in time on task approached, but failed to reach, statistical significance for all three performance tasks. No statistically significant difference was observed in other self-beliefs, course examination performance or course homework performance. These findings align with the overall course findings.

5.1.3.5.2 *None*

Factor specific analysis of the data revealed that students self-identifying as having no programming experience prior to the start of the course, when scaffolded by employment of pair programming in the course laboratories, benefitted overall by a decrease in time on task when compared with peers that did not pair program. No statistically significant difference was observed in other self-beliefs, course examination performance or course homework performance. These findings align with the overall course findings.

5.2 Discussion

Employing the pair programming methodology as an instructional scaffold in the computer laboratory component of the course allowed students to complete the in-laboratory performance tasks in a shorter amount of time with no impact on standard course formative assessment performance (programming homework assignments) or standard course summative assessment performance (programming examinations). This implies that rate of learning course procedural knowledge (i.e., learning how to program) increased when students in the laboratory employed pair programming as a collaborative instructional scaffold. If this implication is correct, and I believe it to be, then we can conclude that the pair programming methodology, when employed as an instructional scaffold, made the learning process observed in this study more efficient.

Employing the pair programming methodology as an instructional scaffold in the computer laboratory component also allowed students to complete the in-laboratory performance tasks with fewer questions of instructors without impact on standard course formative assessment performance or standard course summative assessment performance. This implies that amount of

support required by students while learning course procedural knowledge decreased when students in the laboratory employed pair programming as a collaborative instructional scaffold. If this implication is correct, and I believe it too to be, then we can conclude that the pair programming methodology, when employed as an instructional scaffold, reduced the amount of assistance required by students of instructors during this study, a finding shared by Williams (2001).

5.2.1 Theoretical Underpinnings

The theoretical underpinnings of these results are (1) the procedural knowledge learned by students during completion of the in-laboratory performance tasks lay within their zones of proximal development; (2) students asking questions of instructors, and instructors providing support as a result of those questions, are a traditional form of soft scaffolding; thus, because all students were able to ask questions of instructors while completing their in-laboratory performance tasks, all students received at a minimum this type of instructional scaffolding; (3) pair programming, when used as in the laboratory component of this course, was a type of soft scaffold facilitating articulation and reflection of the unformed procedural knowledge being learned; (4) employment of pair programming supplemented the instructional scaffolding already available to students in the treatment group; these two forms of scaffolding were not mutually exclusive; (5) learning, viewed as the relocation of tasks from the zone of proximal development into the zone of tasks the learner can complete without assistance, occurred in equal amounts between the treatment and control groups; (6) this relocation of tasks required less time when scaffolding of the learning process included pair programming; (7) this relocation of tasks required fewer questions of instructors when scaffolding of the learning process included pair programming.

5.2.2 Differential Impact of the Scaffold

The analytical approach taken in this study to discern differential impact of the scaffold – a series of individual Mann-Whitney U Tests – was selected over alternative methods of analysis due to usage considerations. In particular, I wanted educational practitioners to be able to weigh usage of the pair programming methodology against easily understood, easily identified, discrete factors thus increasing reader generalizability. It was my aim to allow educational practitioners the ability to say "If my class contains a large number of female students, is pair programming likely to be an effective instructional scaffold in my class?" and be able to readily discern an answer from my results. Multiple forms of analysis of variance, as well as factor analysis were considered. However, no alternative would have so clearly revealed the differential impact of the scaffold in the desired light.

Unfortunately, the approach taken to investigating differential impact was hindered in many cases by sample size. For example: those students who performed poorly on the prior programming exam in the course were of particular interest. I had hoped that the instructional scaffold would prove beneficial to them in terms of learning, regardless of impact on those already doing well in the course. Only 9 students in the course fell into this category though, with 4 in the treatment group and 5 in the control. The problem of initially small sample size was compounded by some of these students failing to attend lab (thus further reducing sample size relative to observational data), failing to complete the performance task during the allotted laboratory time (reducing sample size relative to observational data) and failing to submit all homework assignments (thus further reducing sample size relative to this assessment). The findings regarding this group of student are thus quite limited.

Also complicating the findings are the University deadlines that allowed students in the course to drop/withdraw from the class until the end of the 9th week of classes. Recall, this was

after the first laboratory examination and 2 weeks into the treatment phase of the experiment. Anecdotal evidence from students who withdrew suggested that the challenge of learning the course content, both in terms of amount learned and amount of time required learning, were factors in their decision to withdraw. The instructional scaffold may have proven beneficial to these individuals – or not – but in any case their inclusion in the sample, particular in low population demographics, would have allowed us greater confidence in the findings. Relatedly, several studies have found pair programming to aid in student completion rate. Had the treatment (i.e., pair programming) started earlier in the semester, fewer students may have withdrawn from the course.

Many existing studies have concluded that employment of pair programming in the classroom helps students in underrepresented demographics to learn to program (AAUW, 2000; Liebenberg et al., 2012; Werner et al., 2004; Williams & Upchurch, 2001). Because of this, I in particular expected to see differential impact of the scaffold on female students in the course. Unfortunately, the number of female students in the course limited comparative analysis of differential impact. Descriptive statistics suggest no such differential impact occurred for female students in the course though.

Interestingly, a notable differential impact of the scaffold was on students hailing from science disciplines. Pair programming, employed as a scaffold, seemed to have no discernable impact on them at all. Unfortunately, this demographic also suffered from small sample size; the control group consisted of only 4 students. Thus our confidence in these findings is limited. However, should the findings be valid, then pair programming was not an effective scaffold for these students. It also did not hinder these students, as no detrimental impact of its usage was observed. This differential impact is possibly due to these students having employed some alternative scaffold to their learning unavailable to the class at large and unknown to the

researcher. This differential impact could also be due these students having a broader zone of tasks which they can already do unassisted, thus decreasing the relative distance within the zone of proximal development that a task must be relocated during the learning process. For example, science students may have come into the course already in possession of knowledge necessary to computer programming, such as problem decomposition.

Another notable differential impact of the scaffold was on students who performed well on the prior course programming examination. The difference in learning, as reflected in the course performance data, very nearly reaches statistical significance. This implies that those doing well may actually benefit more from the scaffold than those performance is only mediocre or poor. Had the sample size been larger in the study, we are likely to have more confidence in these findings.

Finally, it is important to note that though they had access to all learning materials necessary to complete their weekly homework assignment, and were each week verbally encouraged by laboratory instructors to begin work on their homework assignment, almost all students in the course simply left the computer laboratory after completing their weekly performance task. For example: after completing in-laboratory performance task 06, only three students remained to begin work on their homework (two treatment and one control). We had thought that students would use this time to begin working on their homework assignment, taking advantage of the presence of course instructors while they did so. However, we did not observe this to be true. Valuable laboratory time was thus underutilized.

5.2.3 Support for these Findings in Existing Literature

My review of existing literature located no studies investigating the impact of the pair programming methodology on speed at which learning takes place or efficiency of the learning process. Thus I could find no support in the existing literature for this aspect of my findings. However, this is not surprising given the lack of existing literature investigating the educational usage of pair programming from a learning science perspective.

5.3 Implications for Educational Practitioners

The following list of implications and recommendations is derived from this study, and is provided for educational practitioners.

- Pair programming can be an effective instructional scaffold. However, like all scaffolds, the knowledge to be learned must be appropriately challenging. If the challenge is too small, then no support to the learning process is necessary and the navigator will serve no purpose. If the challenge is too high, then the support to the learning process will be insufficient.
- In a traditional computer laboratory environment, the soft scaffolding technique of student-to-instructor questioning places the burden of instructional support entirely on the instructor. Pair programming, when used appropriately, will distribute a portion of this burden among the students, thus decreasing the instructional burden borne by the instructor.
- When pair programming, the speed at which students learned how to program increased. However, this simply resulted in students leaving the computer laboratory sooner. The amount of material to be learned, or the breadth of the tasks to be completed, should be

increased proportionately if instructors wish to capitalize upon this increase rate of learning.

- Relatedly, if the speed of learning programming procedural knowledge is increased by employing pair programming as a collaborative instructional scaffold, then courses learning objectives can remain constant and the course completed quicker.
- Students from differing academic disciplines have differing zones of proximal development, and thus differing scaffolding needs.

5.4 Recommendations for Future Studies

As with most investigations, as many questions are created as answered. The following list identifies several areas which I believe have potential for further study.

- This study, particularly differential impact of the instructional scaffold on targeted demographics, was hindered by sample size. Would a larger scale study, or a meta-analysis of multiple comparable studies, increase or decrease confidence in our findings?
- The student pairing during this study was random. How, from a learning science perspective, would the effectiveness of pair programming as an instructional scaffold be impacted by such a change?
- Students only spent 3 weeks of the 16 week semester pair programming. How, from a learning science perspective, would the effectiveness of pair programming as an instructional scaffold be impacted by such a change?
- The observer during this study was also the course instructor, who prioritized assisting students over observing them. Observational data was because of this very limited. What more could be learned by video recording students as they work in their natural educational setting?

- Students in the study who hailed from science disciplines displayed no observable impact from the employment of pair programming as an instructional scaffold. Why?
- This study investigated impact of the collaborative scaffold on student discomfort. However, the topic was not the sole focus of the study, and investigation took only the form of questionnaire data. Student discomfort during the learning process, particularly as it relates to quality of student life, warrants more thorough investigation in computing disciplines. Similarly, the impact of a collaborative instructional scaffold on learning discomfort experienced by members of underrepresented demographics, particularly those who believe social isolation an intrinsic component of computing careers, warrants investigation.

5.5 Summary

This chapter interpreted the results provided in the previous chapter, discussing them in context of theory and practice. Implications of these results led to several recommendations for educational practitioners – many of which I wish I had known when I first started teaching. Finally, this chapter ends with the identification of several questions that arose during the course of this study.

LIST OF REFERENCES

LIST OF REFERENCES

American Association Of University Women Education Foundation Commission On Technology, Gender, and Teacher Education. *Tech-Savvy: Educating girls in the new computer age.* (2000). Retrieved from http://www.aauw.org/2000/techsavvy.html

Arisholm, B., Gallis, H., Dyba, T., & Sjoberg, D. I. K. (2007). Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. *IEEE Transactions on Software Engineering*, *33*(2), 65–86. http://doi.org/10.1109/TSE.2007.17

Armstrong, P. (n.d.). Bloom's Taxonomy. Retrieved from https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/

Barker, L., & McGrath Cohoon, J. (2009). *Key Practices for Retaining Undergraduates in Computing*. *Strategic*. Retrieved from http://www.ncwit.org/resources/key-practices-retaining-undergraduates-computing

Beck, K. (2000). *Extreme Programming Explained: Embrace Change* (1st ed.). Addison-Wesley.

Bennedsen, J., & Caspersen, Michael, E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin*, *39*(2), 32–36. http://doi.org/10.1145/1272848.1272879

Bergin, S., & Reilly, R. (2005). The influence of motivation and comfort-level on learning to program. In *Proceedings of the 17th Annual Workshop on the Psychology of Programming Interest Group* (pp. 293–304).

Bornat, R., Dehnadi, S., & Simon. (2008). Mental models, consistency and programming aptitude. In *Proceedings of the tenth conference on Australasian computing education* (pp. 53–61).

Braught, G., Eby, L. M., & Wahls, T. (2008). The effects of pair-programming on individual programming skill. *ACM SIGCSE Bulletin*, *40*(1), 200–205. http://doi.org/10.1145/1352322.1352207

Beed, P., Hawkins, M., & Roller, C. (1991). Moving Learners toward Independence: The Power of Scaffolded Instruction. *Reading Teacher*, *44*(9), 648–655.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (2000). *How people learn : brain, mind, experience, and school.* Washington, D.C.: National Academy Press.

Carver, S. M. (2006). Assessing for Deep Understanding. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 205–221). New York: Cambridge University Press.

Collins, A., Joseph, D., & Bielaczyc, K. (2004). Design Research : Theoretical and Methodological Issues Design Research. *Journal of the Learning Sciences*, *13*(1), 15–42. http://doi.org/10.1207/s15327809jls1301

Confrey, J. (2006). The Evolution of Design Studies as Methodology. In R. K. Sawyer (Ed.), *The Cambridge handbook of the learning sciences* (pp. 135–151). New York: Cambridge University Press.

Connolly, C., Murphy, E., & Moore, S. (2009). Programming Anxiety Amongst Computing Students - A Key in the Retention Debate? *IEEE Transactions on Education*, *52*(1), 52–56. http://doi.org/10.1109/TE.2008.917193

Creswell, J. W. (2008). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (3rd ed.). Los Angeles: Sage Publications.

Creswell, J. W. (2012). *Qualitative Inquiry and Research Design: Choosing Among Five Approaches* (3rd ed.). Los Angeles: Sage Publications.

Cronbach, L. (1975). Beyond the two disciplines of scientific psychology. *American Psychologist*, *30*, 116–127.

DBMS_STAT_FUNCS. (n.d.). In *Oracle Database Online Documentation 11g Release 1 (11.1)*. Oracle Corporation. Retrieved from https://docs.oracle.com/cd/B28359_01/appdev.111/b28419/d_stat_f.htm#i999113

DBRC. (2003). Design-Based Research: An Emerging Paradigm for Educational Inquiry. *Educational Researcher*, *32*(1), 5–8.

Delen, E., Liew, J., & Willson, V. (2014). Effects of interactivity and instructional scaffolding on learning: Self-regulation in online video-based environments. *Computers & Education*, *78*, 312–319. http://doi.org/10.1016/j.compedu.2014.06.018

Erdei, R., Whittinghill, D., & Springer, J. (2014). Collaboration While Programming: Observing Student Perceptions of Pair Programming in the Classroom. In *Proceedings of the World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education* (pp. 543–551). Chesapeake, VA: AACE.

Erdei, R., Whittinghill, D., & Springer, J. (2016). Collaboration While Learning to Program: Investigating the Affective Impact on Students in the Classroom. Manuscript in preparation.

Erdei, R., Whittinghill, D., & Springer, J. & Magana, A. (2016). Collaboration while Programming: Investigating the use of Pair Programming in Mitigating the Impact of Increased Class Size. Manuscript in preparation.

Firestone, W. A. (1993). Alternative Arguments for Generalizing from Data as Applied to Qualitative Research. *Educational Researcher*, *22*(4), 16–23.

Fontana, R. P., Milligan, C., Littlejohn, A., & Margaryan, A. (2015). Measuring Self-Regulated Learning in the Workplace. *International Journal of Training and Development*, *19*(1), 32–52. http://doi.org/10.1111/ijtd.12046

Freeman, S. F., Jaeger, B. K., & Brougham, J. C. (2003). Pair programming: More learning and less anxiety in a first programming course. In *ASEE Annual Conference Proceedings* (pp. 8885–8893). Retrieved from http://www.jonaschalk.neu.edu/search_archives/documents/pairprog03.pdf

Ghasemi, A., & Zahediasl, S. (2012). Normality tests for statistical analysis: a guide for non-statisticians. *International Journal of Endocrinology and Metabolism*, *10*(2), 486–489. http://doi.org/10.5812/ijem.3505

Gos, M. W. (1996). Computer Anxiety and Computer Experience: A New Look at an Old Relationship. *The Clearing House: A Journal of Educational Strategies*, *69*(5), 271–276. http://doi.org/10.1080/00098655.1996.10114315

Jacobson, N. (2009). NCWIT Pair Programming-In-A-Box. Retrieved from www.ncwit.org/pairprogramming

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences* (pp. 53–58).

Johnson, D. W., & Johnson, R. T. (2008). Cooperation and the Use of Technology. In J. M. Spector (Ed.), *Handbook of research on educational communications and technology* (3rd ed., pp. 785–811). New York: Lawrence Erlbaum Associates.

Kubiszyn, T., & Borich, G. (2010). *Educational Testing and Measurement: Classroom Application and Practice* (9th ed.). John Wiley & Sons, Inc.

Latoza, Thomas, D., Venolia, G., & Deline, R. (2006). Maintaining mental models: a study of developer work habits. In *Proceedings of the 28th International Conference on Software Engineering* (pp. 492–501). ACM. http://doi.org/10.1145/1134285.1134355

Liebenberg, J., Mentz, E., & Breed, B. (2012). Pair programming and secondary school girls' enjoyment of programming and the subject Information Technology (IT). *Computer Science Education*, *22*(3), 219–236. http://doi.org/10.1080/08993408.2012.713180

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry*. Beverly Hills, Calif.: Sage Publications.

Linn, M. C. (1995). Designing Computer Learning Environments for Engineering and Computer Science: The Scaffolded Knowledge Integration Framework. *Journal of Science Education and Technology*, *4*(2), 103–126.

Mathison, S. (1988). Why Triangulate? *Educational Researcher*, *17*(2), 13–17.

Mcdowell, C., Werner, L., Bullock, Heather, E., & Fernald, J. (2003). The impact of pair programming on student performance, perception and persistence. In *Proceedings of the 25th International Conference on Software Engineering* (pp. 602–607).

Merriam, S. B. (1988). *Case Study Research in Education. A Qualitative Approach.* Jossey-Bass.

Merriam, S., & Caffarella, R. (2007). *Learning in adulthood. A Comprehensive Guide* (3rd ed.). Jossey-Bass.

Merriam, S., & Caffarella, R. (2007). *Learning in adulthood. A Comprehensive Guide* (Vol. 3). Jossey-Bass.

Miller, P. (2009). *Theories of developmental psychology* (5th ed.). New York: Worth Publishers.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003). Improving the CS1 Experience with Pair Programming. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education* (pp. 359–362). New York, NY, USA: ACM. http://doi.org/10.1145/792548.612006

Nagappan, N., Williams, L., Wiebe, E., Miller, C., Balik, S., Ferzli, M., & Petlick, J. (2003). Pair Learning: With an Eye Toward Future Success. In *Extreme Programming and Agile Methods — {XP/Agile} Universe 2003* (Vol. 2753, pp. 185–193). Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-540-45122-8_21

National Research Council, (2002). *Scientific research in education*. (R. J. Shavelson; & L. Towne, Eds.). Washington, DC: National Academy Press.

Palinscar, A. S., & Brown, A. L. (1984). Reciprocal Teaching of Comprehension-Fostering and Comprehension-Monitoring Activities. *Cognition and Instruction*, *1*(2), 117–175.

Pallant, J. (2010). *SPSS Survival Manual a Step by Step Guide to Data Analysis Using SPSS*. (4th ed.). Maidenhead : McGraw-Hill International UK Ltd.: McGraw-Hill International UK Ltd.

Pea, R. D. (2004). The Social and Technological Dimensions of Scaffolding and Related Theoretical Concepts for Learning, Education, and Human Activity. *Journal of the Learning Sciences*, *13*(3), 423–451.

Popham, W. J. (2010). *Classroom Assessment: What Teachers Need To Know* (6th ed.). Boston: Pearson.

Purdue Majors and Minors. (n.d.). Retrieved from http://www.admissions.purdue.edu/majors/

Purdue University myPurdue Self-Service Catalog Entries: CNIT 17500. (n.d.). Retrieved from https://selfservice.mypurdue.purdue.edu/

Purdue University West Lafayette Enrollment Summary: Fall 2015. (2015). Retrieved from http://www.purdue.edu/enrollmentmanagement/researchanddata/enrollment-summary/2015/fall/PWL.pdf

Quintana, C., Reiser, B. J., Davis, E. A., Krajcik, J., Fretz, E., Duncan, R. G., … Soloway, E. (2004). A Scaffolding Design Framework for Software to Support Science Inquiry. *The Journal of the Learning Sciences*, *13*(3), 337–386. http://doi.org/10.1207/s15327809jls1303

Rahman, M. M., & Govindarajulu, Z. (1997). A modification of the test of Shapiro and Wilk for normality. *Journal of Applied Statistics*, *24*(2), 219–236. http://doi.org/10.1080/02664769723828

Rosenshine, B., & Meister, C. (1992). The Use of Scaffolds for Teaching Higher-Level Cognitive Strategies. *Educational Leadership*, *49*(7), 26–33.

Ryan, R. M., & Connell, J. P. (1989). Perceived Locus of Causality and Internalization: Examining Reasons for Acting in Two Domains. *Journal of Personality and Social Psychology*, *57*(5), 749–761. http://doi.org/10.1037/0022-3514.57.5.749

Sawyer, R. K. (2006). *The Cambridge handbook of the learning sciences*. Cambridge ; New York: Cambridge University Press.

Sax, L. J. (2112). *Examining the Underrepresentation of Women in STEM Fields: Early Findings from the Field of Computer Science*. Retrieved from http://www.escholarship.org/uc/item/84j9s1v1

Schneider, M., & Stern, E. (2010). The Developmental Relations between Conceptual and Procedural Knowledge: A Multimethod Approach. *Developmental Psychology*, *46*(1), 178–192. http://doi.org/10.1037/a0016701

Schulz, M., & Rosznagel, C. S. (2010). Informal Workplace Learning: An Exploration of Age Differences in Learning Competence. *Learning and Instruction*, *20*(5), 383–399. http://doi.org/10.1016/j.learninstruc.2009.03.003

Scott, M. J., & Ghinea, G. (2014). Measuring enrichment: the assembly and validation of an instrument to assess student self-beliefs in CS1. In *Proceedings of the tenth annual conference on International computing education research* (pp. 123–130). New York, New York, USA: ACM. http://doi.org/10.1145/2632320.2632350

Shapiro, S. S., & Wilk, M. B. (1965). An Analysis of Variance Test for Normality (Complete Samples). *Biometrika*, *52*(3/4), 591–611. http://doi.org/10.2307/2333709

Slavin, R. E. (1987). Developmental and motivational perspectives on cooperative learning; a reconciliation. *Child Development*, *58*(5), 1161–1167.

Speier, C., Morris, M. G., & Briggs, C. M. (1995). Attitudes Toward Computers: The Impact on Performance. In *Americas Conference on Information Systems*.

STATS_ONE_WAY_ANOVA. (n.d.). In *Oracle Database Online Documentation 11g Release 1 (11.1)*. Oracle Corporation. Retrieved from http://docs.oracle.com/cd/B28359_01/server.111/b28286/functions163.htm

STATS_T_TEST_*. (n.d.). In *Oracle Database Online Documentation 11g Release 1 (11.1)*. Oracle Corporation. Retrieved from http://docs.oracle.com/cd/B28359_01/server.111/b28286/functions164.htm#SQLRF06323

Summers, J., Gorin, J., Beretvas, S., & Svinicki, M. (2005). Evaluating Collaborative Learning and Community. *The Journal of Experimental Education*, *73*(3), 165–188. http://doi.org/10.3200/JEXE.73.3.165-188

Tobias, S. (1982). When Do Instructional Methods Make a Difference? *Educational Researcher*, *114*(4), 4–9.

Vygotsky, L. (1978). *Mind in society : the development of higher psychological processes*. Cambridge, MA: Harvard University Press.

Walker, R. (1980). The conduct of educational case studies: Ethics, theory and procedures. In *Rethinking educational research* (pp. 30–63).

Wang, F., & Hannafin, M. J. (2005). Design-based research and technology-enhanced learning environments. *Educational Technology Research & Development*, *53*(4), 5–23. http://doi.org/10.1007/BF02504682

Watson, C., & Li, F. (2014). Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 39–44). ACM. http://doi.org/10.1145/2591708.2591749

Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *Journal on Educational Resources in Computing*, *4*(1), 4–es. http://doi.org/10.1145/1060071.1060075

Wilfong, J. D. (2006). Computer anxiety and anger: the impact of computer use, computer experience, and self-efficacy beliefs. *Computers in Human Behavior*, *22*(6), 1001–1011.

Williams, L. (2006). Debunking the nerd stereotype with pair programming. *Computer*, *39*(5), 83–85. http://doi.org/10.1109/MC.2006.160

Williams, L., & Upchurch, R. L. (2001). In support of student pair-programming. In *SIGCSE '01 Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education* (Vol. 33, pp. 327–331). http://doi.org/10.1145/366413.364614

Wood, D., Bruner, J. S., & Ross, G. (1976). The Role OF Tutoring In Problem Solving. *Journal of Child Psychology and Psychiatry*, *17*(2), 89–100. http://doi.org/10.1111/j.1469-7610.1976.tb00381.x

Wood, D., & Wood, H. (1996). Vygotsky, Tutoring and Learning. *Oxford Review of Education*, *22*(1), 5–16.

Yelland, N., & Masters, J. (2007). Rethinking Scaffolding in the Information Age. *Computers and Education*, *48*(3), 362–382. http://doi.org/10.1016/j.compedu.2005.01.010

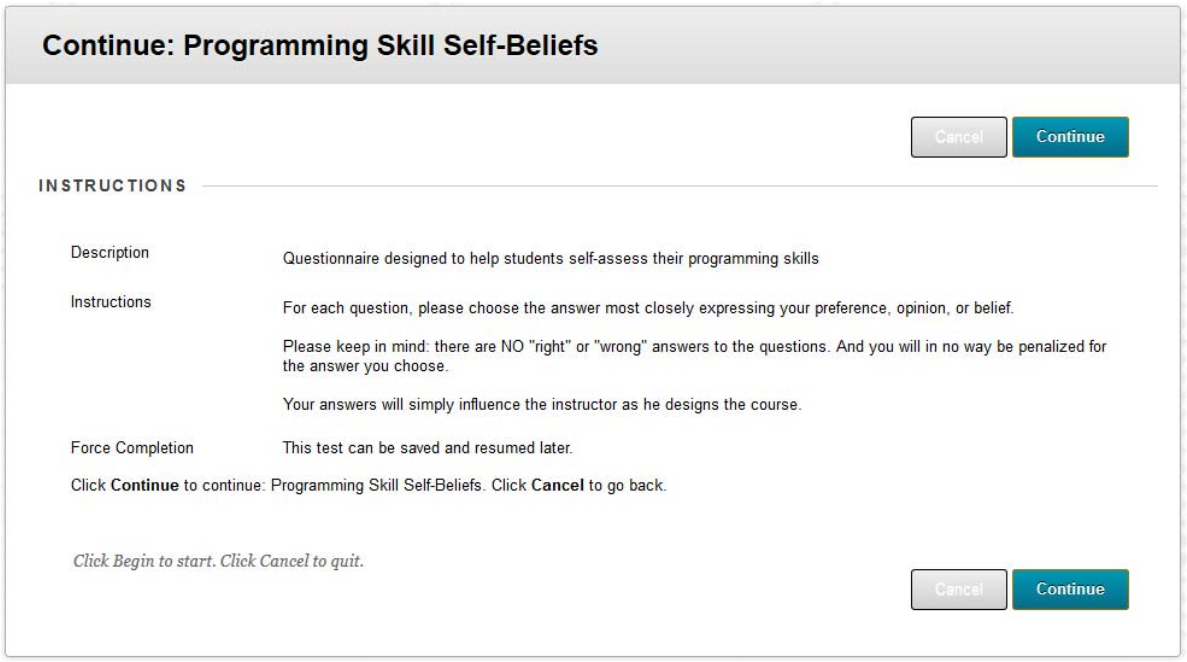
Yin, R. K. (1984). *Case study research : design and methods* (2nd ed.). Beverly Hills, Calif: Sage Publications.

APPENDICES

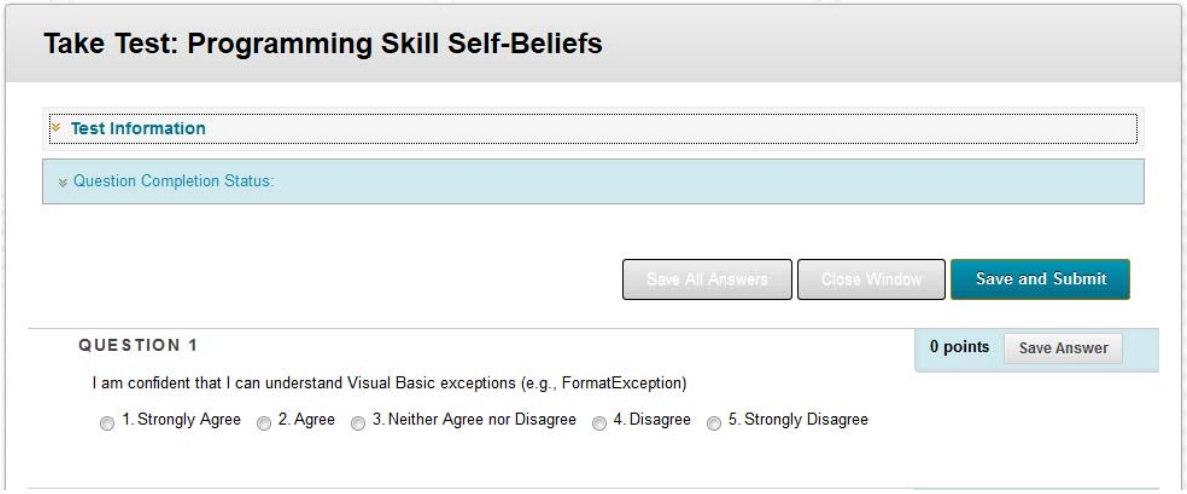
Appendix A. Questionnaire on Self Beliefs

Administered to: All students as Pre-Test and Post-Test

Instructions provided to students before beginning the questionnaire



Questions



QUESTION 2 — 0 points — Save Answer

I am confident I can solve simple problems with my programs

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 3 — 0 points — Save Answer

I am confident I can implement a method from a description of a problem or algorithm

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 4 — 0 points — Save Answer

I am confident I can debug a program that calculates prime numbers

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 5 — 0 points — Save Answer

I am just not good at programming

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 6 — 0 points — Save Answer

I learn programming quickly

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 7 — 0 points — Save Answer

I have always believed that programming is one of my best subjects

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 8 — 0 points — Save Answer

In my programming labs, I can solve even the most challenging problems

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 9 — 0 points — Save Answer

I enjoy reading about programming

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 10 — 0 points — Save Answer

I do programming because I enjoy it

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 11 — 0 points — Save Answer

I am interested in the things I learn in programming classes

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 12 — 0 points — Save Answer

I think programming is interesting

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 13 — 0 points — Save Answer

I often worry that it will be difficult for me to complete debugging exercises

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 14 — 0 points — Save Answer

I often get tense when I have to debug a program

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 15 — 0 points — Save Answer

I get nervous when trying to solve programming bugs

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 16 — 0 points — Save Answer

I feel helpless when trying to solve programming bugs

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 17 — 0 points — Save Answer

I have a fixed level of programming aptitude, and not much can be done to change it

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 18 — 0 points — Save Answer

I can learn new things about software development, but I cannot change my basic aptitude for programming

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 19 — 0 points — Save Answer

To be honest, I do not think I can really change my aptitude for programming

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

Click Save and Submit to save and submit. Click Save All Answers to save all answers.

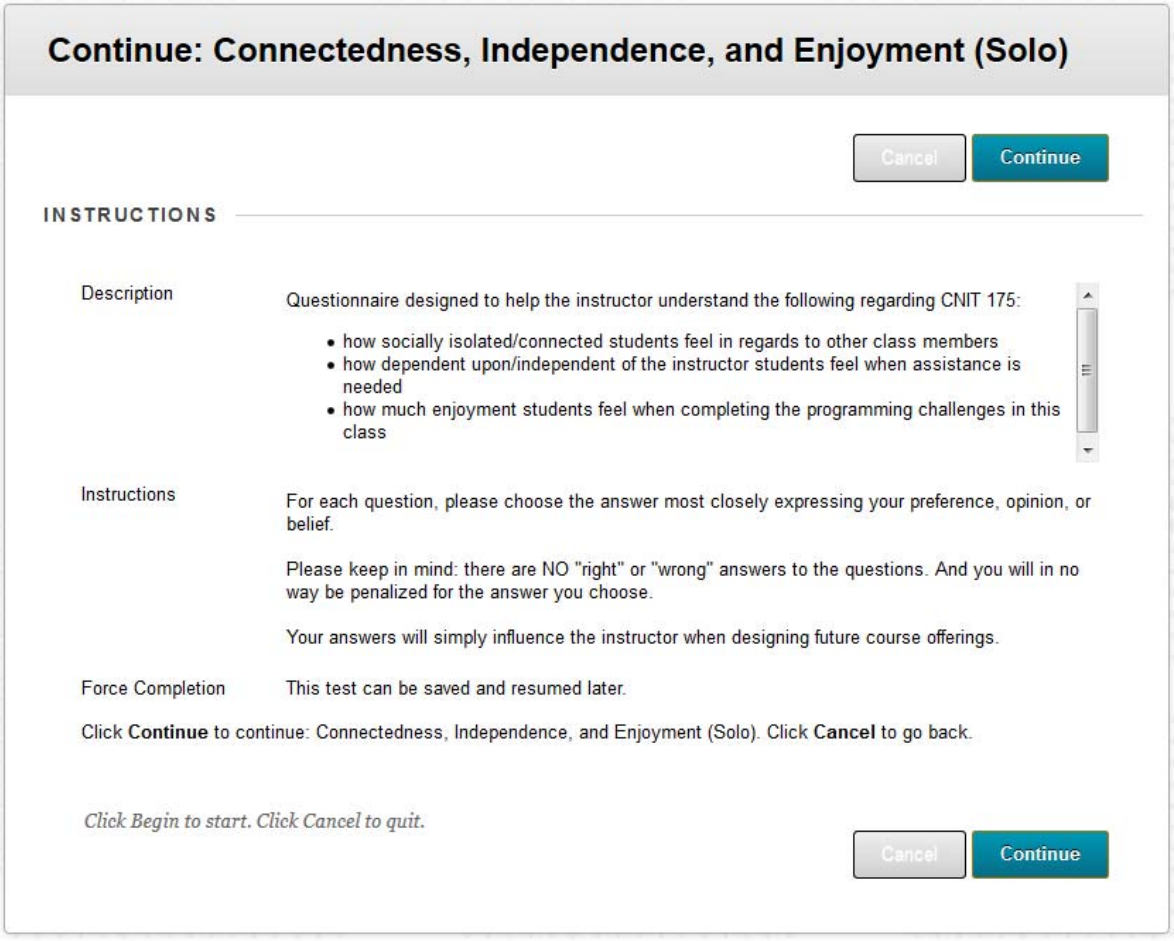
Save All Answers | Close Window | Save and Submit

Appendix B. Questionnaire on Sense of Connectedness, Community, and Instructor Independence

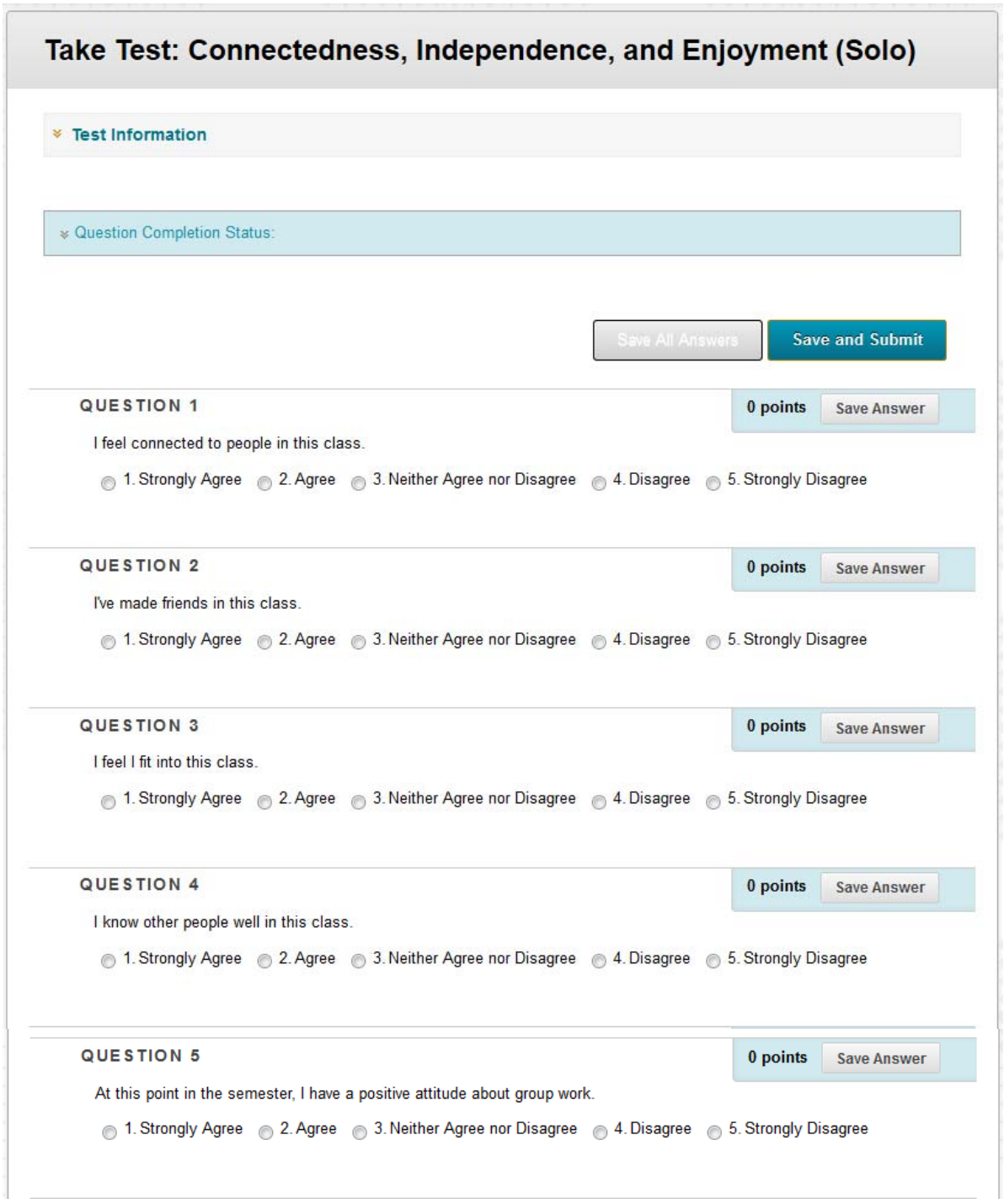
(Solo version)

Administered to: All students as Pre-Test, Control Group as Post-Test

Instructions provided to students before beginning the questionnaire



Questions



Take Test: Connectedness, Independence, and Enjoyment (Solo)

Test Information

Question Completion Status:

Save All Answers | Save and Submit

QUESTION 1 — 0 points — Save Answer

I feel connected to people in this class.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 2 — 0 points — Save Answer

I've made friends in this class.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 3 — 0 points — Save Answer

I feel I fit into this class.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 4 — 0 points — Save Answer

I know other people well in this class.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 5 — 0 points — Save Answer

At this point in the semester, I have a positive attitude about group work.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 6 — 0 points — Save Answer

I enjoy doing my IN Lab programming assignments.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 7 — 0 points — Save Answer

I enjoy doing my POST Lab programming assignments.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 8 — 0 points — Save Answer

I enjoy solving difficult programming problems.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 9 — 0 points — Save Answer

I try hard to do well in school.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 10 — 0 points — Save Answer

Prior to coming to lab, I adequately prepare for the programming assignment.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 11 — 0 points — Save Answer

Additional preparation on my part would have made the laboratory programming assignments easier.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 12 — 0 points — Save Answer

In spite of the programming assignment, I actually enjoy myself during lab.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 13 — 0 points — Save Answer

Completing a laboratory programming assignment provides me with a sense of satisfaction.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 14 — 0 points — Save Answer

I enjoyed myself while completing the laboratory programming assignments.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 15 — 0 points — Save Answer

While completing the laboratory programming assignment, I required little assistance from the instructor.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 16 — 0 points — Save Answer

During the laboratory, an instructor was available to provide assistance when I required it.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 17 — 0 points — Save Answer

In lab, if I needed assistance from an instructor I had to wait a long time.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

Click Save and Submit to save and submit. Click Save All Answers to save all answers.

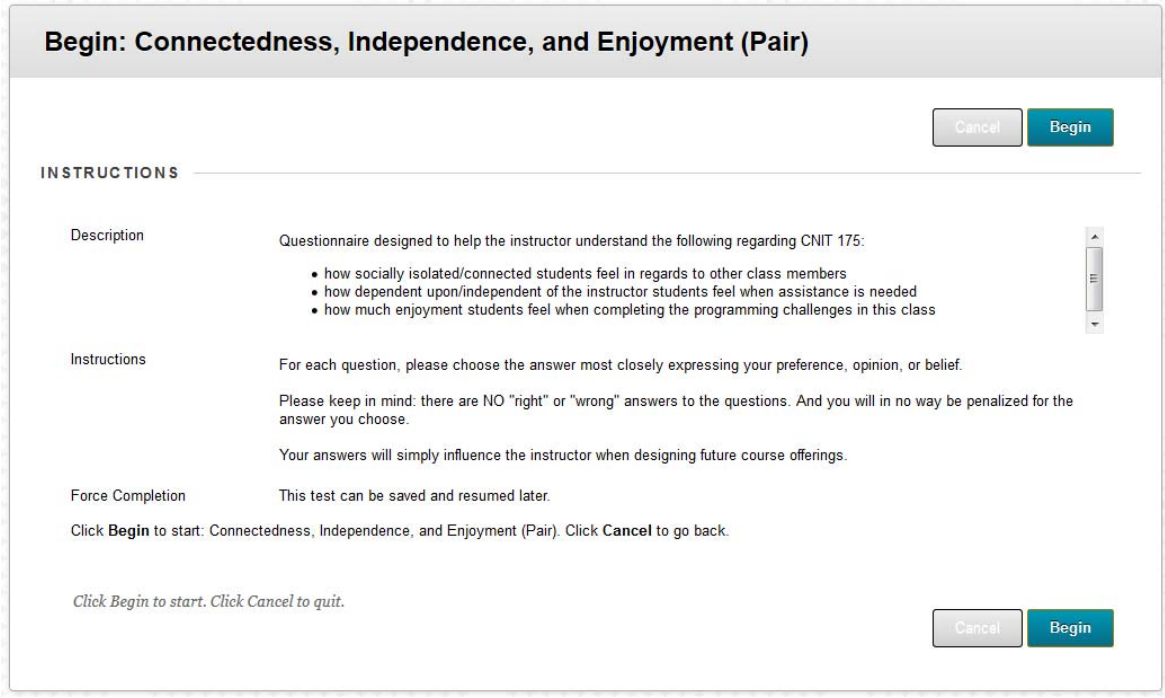
Save All Answers | Save and Submit

Appendix C. Questionnaire on Sense of Connectedness, Community, and Instructor Independence

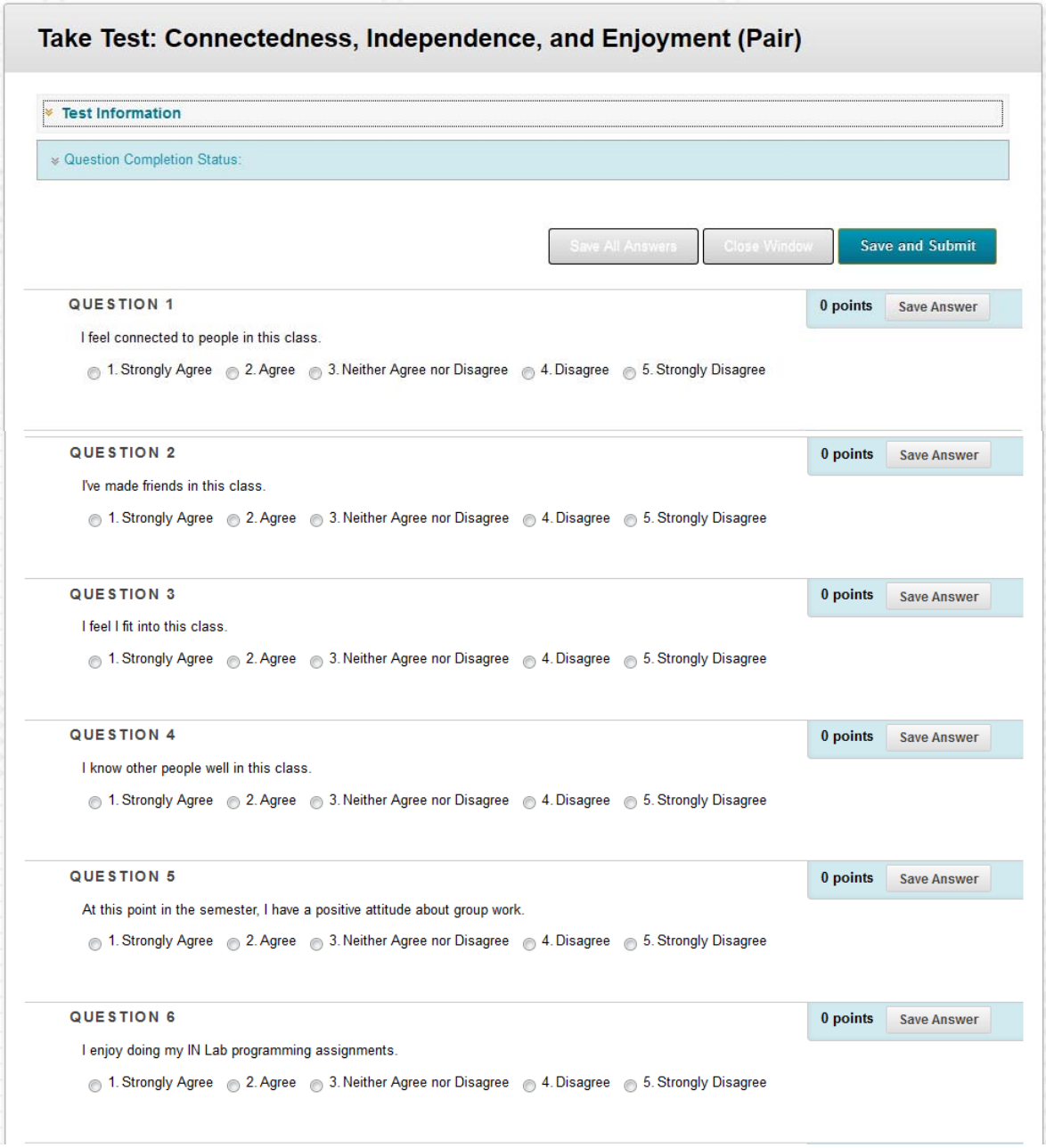
(Group version)

Administered to: Experimental Group as Post-Test

Instructions provided to students before beginning the questionnaire



Questions



Take Test: Connectedness, Independence, and Enjoyment (Pair)

Test Information

Question Completion Status:

Save All Answers | Close Window | Save and Submit

QUESTION 1 — 0 points — Save Answer

I feel connected to people in this class.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 2 — 0 points — Save Answer

I've made friends in this class.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 3 — 0 points — Save Answer

I feel I fit into this class.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 4 — 0 points — Save Answer

I know other people well in this class.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 5 — 0 points — Save Answer

At this point in the semester, I have a positive attitude about group work.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 6 — 0 points — Save Answer

I enjoy doing my IN Lab programming assignments.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 7 — 0 points — Save Answer

I enjoy doing my POST Lab programming assignments.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 8 — 0 points — Save Answer

I enjoy solving difficult programming problems.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 9 — 0 points — Save Answer

I try hard to do well in school.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 10 — 0 points — Save Answer

Prior to coming to lab, I adequately prepare for the programming assignment.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 11 — 0 points — Save Answer

Additional preparation on my part would have made the laboratory programming assignments easier.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 12 — 0 points — Save Answer

In spite of the programming assignment, I actually enjoy myself during lab.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 13 — 0 points — Save Answer

Completing a laboratory programming assignment provides me with a sense of satisfaction.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 14 — 0 points — Save Answer

I enjoyed myself while completing the laboratory programming assignments.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 15 — 0 points — Save Answer

While completing the laboratory programming assignment, I required little assistance from the instructor.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 16 — 0 points — Save Answer

During the laboratory, an instructor was available to provide assistance when I required it.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 17 — 0 points — Save Answer

In lab, if I needed assistance from an instructor I had to wait a long time.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 18 — 0 points — Save Answer

My partner and I spent equal time in each role (i.e., equal time as driver, and equal time as navigator).

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 19 — 0 points — Save Answer

When my partner was in the navigator role, they actively assisted me with the assignment.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 20 — 0 points — Save Answer

When I was in the navigator role, my partner accepted my assistance.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 21 — 0 points — Save Answer

While completing the exercise, there were times I felt uncomfortable because of my partner.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 22 — 0 points — Save Answer

I felt at ease working with my partner.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 23 — 0 points — Save Answer

My partner made me feel uncomfortable.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 24 — 0 points — Save Answer

My partner was sufficiently prepared for the laboratory exercises.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 25 — 0 points — Save Answer

While completing the laboratory exercises, my partner's assistance was very helpful.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 26 — 0 points — Save Answer

My partner contributed very little while completing the exercise.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 27 — 0 points — Save Answer

It was important to me that my laboratory partner learns the material we completed together.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 28 — 0 points — Save Answer

My laboratory partner's success in the course was important to me.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 29 — 0 points — Save Answer

I felt, at least in part, responsible for my laboratory partner's learning of the course material.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 30 — 0 points — Save Answer

My success in the course was important to my laboratory partner.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 31 — 0 points — Save Answer

My laboratory partner seemed to feel, at least in part, responsible for my learning of the course material.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 32 — 0 points — Save Answer

Overall, each of the group members contributed his or her share.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 33 — 0 points — Save Answer

Overall, my group was effective working together.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 34 — 0 points — Save Answer

Typically, my group had a clear understanding of the expectations for the group tasks.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 35 — 0 points — Save Answer

Overall, my group members responded positively to peer questions.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 36 — 0 points — Save Answer

Typically, most group members shared their own ideas during group work.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 37 — 0 points — Save Answer

My group was successful in completing the requirements of most tasks.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 38 — 0 points — Save Answer

I value my group as a resource for learning.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 39 — 0 points — Save Answer

As a result of group work, I improved my group-building skills.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

QUESTION 40 — 0 points — Save Answer

As a result of group work, I improved my problem-solving skills.

1. Strongly Agree 2. Agree 3. Neither Agree nor Disagree 4. Disagree 5. Strongly Disagree

Click Save and Submit to save and submit. Click Save All Answers to save all answers.

Save All Answers | Close Window | Save and Submit

VITA

VITA

Ronald Erdei

Education

Ph.D. candidate	Computer and Information Technology Purdue University, West Lafayette IN 47906	August 2016 (expected)
M.S.	Computer and Information Technology Purdue University, West Lafayette IN 47906	2006
B.S.	General Sciences: Biology, Chemistry Purdue University, West Lafayette IN 47906	1994

Research Interests

My research interests lie in the learning sciences, and serve to compliment my teaching interests. Specific topics of interest include: instructional scaffolding in computing disciplines, cooperative learning in college students, pedagogical practices aimed at reducing student discomfort during the learning process, optimization of content delivery for targeted populations (particularly non-traditional college students).

Teaching Experience

Instructor of Record

In all cases, taught lecture component of the course. In some cases, taught computer laboratory component of the course as well. Assisted individual students to overcome course-related challenges using face-to-face, email, and video formats. Developed all learning activities in relation to pre-defined course learning objectives. Developed all assessment mechanisms to evaluate student mastery of learning objectives. Administered all components of the course via a centralized course management system (Blackboard). Supervised part time employees (graders, teaching assistants).

Course	Enrollment	Semester/Year
CNIT 175: Introduction to Visual Programming Format: Traditional (Lecture: 2 Lab: 1)	84	Spring 2016
CNIT 175: Introduction to Visual Programming Format: Traditional (Lecture: 2 Lab: 1)	60	Fall 2015

CNIT 175: Introduction to Visual Programming Format: Distance	21	Summer 2015
CNIT 175: Introduction to Visual Programming Format: Traditional (Lecture: 2 Lab: 1)	114	Spring 2015
CNIT 372: Database Programming Format: Traditional (Lecture: 2 Lab: 1)	61	Fall 2014
CNIT 175: Introduction to Visual Programming Format: Hybrid Distance/Traditional	18	Summer 2014
CNIT 372: Database Programming Format: Traditional (Lecture: 2 Lab: 1)	41	Fall 2013
CNIT 272: Database Fundamentals Format: Traditional (Lecture: 2 Lab: 1)	45	Spring 2013
CNIT 392: Enterprise Data Management Format: Traditional (Lecture: 2 Lab: 1)	33	Spring 2013
CNIT 272: Database Fundamentals Format: Traditional (Lecture: 2 Lab: 1)	69	Fall 2012
CNIT 255: Programming for the Internet Format: Traditional (Lecture: 2 Lab: 1)	92	Fall 2012
CNIT 255: Programming for the Internet Format: Traditional (Lecture: 2 Lab: 1)	38	Spring 2012
CNIT 372: Database Programming Format: Traditional (Lecture: 2 Lab: 1)	19	Spring 2009

Teaching Assistant / Laboratory Instructor

Taught computer laboratory component of the course. Assisted individual students to overcome course-related challenges using face-to-face, email, and video formats. Developed learning activities and assessment mechanisms. Supervised part time employees (graders, teaching assistants).

Course	Enrollment	Semester/Year
CNIT 175: Introduction to Visual Programming Format: Traditional (Lecture: 2 Lab: 1)	89	Fall 2014
CNIT 175: Introduction to Visual Programming Format: Traditional (Lecture: 2 Lab: 1)	119	Spring 2014
CNIT 255: Programming for the Internet Format: Traditional (Lecture: 2 Lab: 1)	19	Spring 2011

Teaching Certifications

Graduate Teaching Certificate, Purdue University Center for Instructional Excellence — Spring 2016

Teaching Awards

Graduate Teaching Award, Purdue University Teaching Academy Spring 2016

Instructional Design Experience

CNIT 175: Introduction to Visual Programming
Migrated CNIT 175 first from a traditional format to a hybrid format, then to a full distance learning format. All formats shared learning objectives with the traditionally taught version of the course yet employed pedagogical strategies, instructional activities, instructional scaffolds, and assessment mechanisms appropriate to the distance format.

Publications

Peer Reviewed Articles in Conference Proceedings

Erdei, R., Whittinghill, D. & Springer, J. (2014). Collaboration While Programming: Observing Student Perceptions of Pair Programming in the Classroom. In *Proceedings of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education* 2014 (pp. 543-551). Chesapeake, VA: Association for the Advancement of Computing in Education (AACE). Retrieved May 23, 2016 from https://www.learntechlib.org/p/148770.

Erdei, R., & Lutes, K. (2012). Exploring Our Options: Modern Publishing Alternatives for our Computer Programming Textbook. *In Proceedings of Informing Science & IT Education Conference* (InSITE) (pp. 223 – 234). Montreal, Quebec, Canada: Informing Science Institute.

Erdei, R., & Slazinski, E. (2004). Characteristics of Streaming Data Acquisition Management Systems. In *Proceedings for The 8th World Multi-Conference on Systemics, Cybernetics, and Informatics* (pp. 283 – 288). Orlando.

Authored Textbooks

Lutes, K., Harriger, A., & Erdei, R. (2011). *An Information Systems Approach to Object-Oriented Programming* (version 2.1.). Self-Published.

Poster Presentations

Erdei, R., Whittinghill, D., & Springer, J. (2015). *Collaboration While Programming: Observing Student Perceptions of Pair Programming in the Classroom*. Poster session at the Purdue University Teaching Academy Day (September 2015).

Erdei, R., Springer, J., & Whittinghill, D. (2014). *The Affective Impact of Pair Programming on Students Enrolled in a College Junior-Level Database Programming Course*. Poster session at the Purdue University Computer and Information Technology Undergraduate Research Open House, West Lafayette, IN (September 2014).

Erdei, R., Springer, J., & Whittinghill, D. (2014). *The Affective Impact of Pair Programming on Students Enrolled in a College Junior-Level Database Programming Course*. Poster session at the Purdue College of Technology Faculty Convocation 10th Annual Poster Session, West Lafayette, IN (March, 2014).

Industry Experience

Database Administrator 2008 - 2011
Information Technology at Purdue (ITaP), Purdue University

- Administered approximately 350 SQL Server and 75 Oracle databases as a member of a 7-person database administration team.
- Increased database security via continual investigation, threat assessment, resource (data) discovery, resource (data) classification, surface area reduction, and adherence to the principle of least privilege.
- Established departmental data –access best practices, helped define new database naming / coding standards, and championed the refining of existing data authorization practices.
- Mentored developers, programmers, and other clients upon database-specific developmental best practices, including secure programming practices.

Database Architect / Analyst / Programmer 2005 - 2008
Housing and Food Services (HFS), Purdue University

- Designed, developed, refactored, and supported database storage objects upon both SQL Server and Oracle database management systems.
- Designed, developed, refined, and supported client specific applications utilizing Microsoft technologies (SQL Server, C# .NET) in a team oriented environment.
- Developed custom reports for clients leveraging various data resources (SQL Server, Oracle, Sybase, Access, SAP, Cognos) as appropriate to their individual business models.
- Documented new and existing processes.
- Investigated, assessed, and subsequently provided recommendations upon tools and technologies for possible use by peers and clients.
- Mentored others in relational data store design and relational programmatic logic.

Database Analyst 2004 - 2005
Walker Information

- Designed, developed, refined, and supported client specific databases using the Oracle DBMS.
- Loaded, scrubbed, merged, validated and subsequently populated

databases with data sets of varying sizes and from various sources.
- Enforced client specific business rules using procedures, triggers, functions, and packages.
- Communicated with clients (both internal and external) using language appropriate to their level of technical expertise.

Information Technology Skills

Programming Languages:	SQL, T-SQL, PL/SQL, C#, Visual Basic, HTML
Database Platforms:	Microsoft SQL Server, Oracle
Tools:	Microsoft Visual Studio, Oracle SQL Developer, Oracle Enterprise Manager

Information Technology Certifications

Information Technology Infrastructure Library (ITIL) v3 Foundation in IT Service Management Certification

Service to the Institution and Profession

Departmental

Instructor, TECSUP Study Abroad Program — Fall 2013
Created, implemented, and administered a 2-hour module on database security for approximately 22 undergraduate students from Peru's Tecnologia Superior University (TECSUP) that participated in an exchange program with Purdue University.

Instructor, TECSUP Study Abroad Program — Fall 2011
Created, implemented, and administered a 2-hour module on database security for approximately 18 undergraduate students from Peru's Tecnologia Superior University (TECSUP) that participated in an exchange program with Purdue University.

College

Marshal Assistant, Purdue University Commencement — Spring 2016
Assisted with the coordination and monitoring of approximately 300 College of Technology / Polytechnic Institute undergraduate students participating in the Spring 2016 commencement.

Instructor, Science Bound Program's Junior Shadow Day — Spring 2014
Created, implemented, and administered a 3 hour session introducing 13 high school students from the Indianapolis Public School system to careers within information technology (IT) disciplines. Extrapolated beyond simple job titles and annual salaries via discussion with live IT practitioners

regarding what they actually do, day to day. Designed a hands-on component in which students created a computer application as a means of demystifying application development in general and information technology as a whole.

University

Presenter (2 sessions), BoilerWeb 2010 Intra-University Conference on Web Technologies. Presentations: (1) *SQL Injection: What It Is & Why It Matters To You* and (2) *Database Concepts: Objects, Owners, and Indexes*. — March 2010

Academic Memberships

Technology Graduate Student Advisory Committee (TGSAC) — 2011-2015
Member: 4 years. Co-Chairperson: 2.5 years.

Professional Memberships

Association for Computing Machinery (ACM) — 2013 - present
Member (student): 3 years.

ACM Special Interest Group for Information Technology Education (SIGITE) — 2013 - present
Member (student): 3 years.

References

David M. Whittinghill, Associate Professor
Department of Computer Graphics Technology, Purdue University
Email Address: dmwhittinghill@purdue.edu
Professional Biography: https://polytechnic.purdue.edu/profile/davewhit

John Springer, Associate Professor and Assistant Department Head
Department of Computer and Information Technology, Purdue University
Email Address: ja@purdue.edu
Professional Biography: https://polytechnic.purdue.edu/profile/jaspring

Marcus Rogers, Professor/Department Head
Department of Computer and Information Technology, Purdue University
Email Address: rogersmk@purdue.edu
Professional Biography: https://polytechnic.purdue.edu/profile/rogersmk