VOICES OF WOMEN IN A SOFTWARE ENGINEERING COURSE: REFLECTIONS ON COLLABORATION

Sarah B. Berenson[1], Kelli M. Slaten[1], Laurie Williams[2], and Chih-wei Ho[2]

[1]Center for Research in Mathematics and Science Education and [2]Computer Science

North Carolina State University

Those science, mathematics, and engineering faculty who are serious about making the education they offer as available to their daughters as to their sons are, we posit, facing the prospect of dismantling a large part of its traditional pedagogical structure, along with the assumptions and practice which support it. [Seymour and Hewett, 1997]

Abstract

Prior research has indicated that female students can be concerned about the insularity of working alone for long periods of time, as they perceive to be the case with computer science education and Information Technology careers. We studied an advanced undergraduate software engineering course at North Carolina State University to characterize the potential of the collaborative learning environments created via pair programming and agile software development to ameliorate these concerns. A collective case study of three representative females in the course revealed four common themes: working with others; productivity; confidence; and interest in IT careers. Three conjectures concerning collaboration emerged from our study. These conjectures included the importance of face-to-face meetings, an increased confidence among women based on product quality, and a reduction in the amount of time spent on assignments. Additionally, we propose a model for future testing that connects these three factors with an increased interest in IT careers.

Categories and Subject Descriptors:

D.2 SOFTWARE ENGINEERING

K.3 COMPUTERS AND EDUCATION

General Terms: Human Factors

Additional Key Words and Phrases: agile software development, collaboration, pair programming, active learning, pedagogy, collaborative learning

1 Introduction

Traditionally, computer science and software engineering education has emphasized working alone, whereby collaboration is cheating. Alternately, larger projects involve “divide-and-conquer” teamwork, in which the team members are assigned roles (such as team leader, development manager, or tester) and break up the whole project into relatively independent parts that each team member can work on primarily alone. In this paper, we share experiences with collaborative pedagogical techniques that we have found can increase the success and satisfaction of female students. Prior studies, including [Margolis, 1997], have indicated that female students can be concerned about the insularity of working alone for long periods of time, as they perceive to be the case with computer science education and Information Technology (IT) careers. As a result, we have initiated a study of pedagogical techniques that emphasize collaboration and teamwork over working alone. In this report, we examine the impression of female students on these techniques.

A longitudinal study has been initiated in the software engineering course at North Carolina State University (NCSU). When students take this course, they are in their third and fourth year of study and have completed six computer science courses. The course involves the development of several smaller programs and one larger team project. Instead of having the students work primarily alone, in the study the students work in pairs for two out of three assignments, employing the pair programming technique [Williams and Kessler, 2003], or in collaborative team settings which employ agile software development techniques [Cockburn, 2001]. To examine the impressions of this substantial pedagogical change, we conducted an in-depth inquiry of three representative females in the class. Specifically, we conducted personal interviews [Berenson, Slaten et al., 2004] with the three young women and examined a retrospective paper each prepared as a course assignment. *The goal of our study was to determine if the use of pair programming and collaborative aspects of agile software development is a favorable change from the female perspective.*

2 Background

In this section, we discuss pair programming and agile software development. From a general education perspective, we provide background research of gender studies, and the frameworks and questions guiding this study.

2.1 Pair Programming

Pair programming refers to the practice whereby two programmers work together at one computer, collaborating on the same design, algorithm, code, or test. The pair is made up of a *driver,* who actively types at the computer or records a design; and a *navigator*, who watches the work of the driver and attentively identifies problems, asks clarifying questions, and makes suggestions. Both are also continuous brainstorming partners. Pair programming has been sporadically used by students and professionals for decades. The use of the practice has grown in the last five years due to its inclusion as a prominent software development practice of the agile software development methodology, Extreme Programming (XP) [Beck, 2000].

Computer science faculty are increasingly experimenting, either formally or informally, with pair programming in the classroom. Initial research results [Cockburn and Williams, 2000; Williams, Kessler et al., 2000; Williams, 2000] indicate that pair programmers produce higher quality code in about half the time of solo programmers. These findings are based on experiments conducted at the University of Utah in a senior-level software engineering course. The focus of that research was the affordability of pair programming and the ability of the practice to yield higher quality code without significant increases in time or cost. The researchers also observed educational benefits for student pair programmers. These benefits include higher scores on graded assignments, increased satisfaction and confidence, reduced student frustration, and reduced workload for the teaching staff.

These observations inspired further research examining the use of pair programming in educating computer science students. Educators at the University of California at Santa Cruz (UCSC) [Bevan, Werner et al., 2002; McDowell, Werner et al., 2002; McDowell, Werner et al., 2003] and NCSU [Williams, Wiebe et al., 2002; Williams, Yang et al., 2002; Nagappan, Williams et al., 2003] have experimented with pair programming in introductory undergraduate programming courses with over 1200 first-year students. These researchers observed that pair programming has a positive impact on multiple aspects of student performance and enjoyment; none of the findings suggest that student learning is compromised.

The UCSC study performed targeted research on the success and retention of female students in introductory classes, comparing those who pair programmed and those who solo programmed. Their findings suggest that women who pair programmed had a higher average programming score (86.9% vs. 70.1%), were more confident in their solutions, and were more likely to have chosen a computer science-related major (59.5% vs. 22.2%).

2.2 Agile Software Development

Software development, where the bulk of IT development falls, can be described using process models. These models can be characterized as plan-driven or agile [Boehm, 2002]. The plan-driven models have an implicit assumption that a good deal of information about the product being developed can be obtained up front. As a result, creating a plan for the project to follow is advisable. An overriding philosophy of plan-driven software models is that the cost of product development can be minimized by creating detailed plans and by constructing and inspecting architecture and design documents prior to initiating code development. As a result of these activities, team members often "talk" through documents rather than face-to-face. Until relatively recently, plan-driven methodologies were most often taught in universities as formalized ways of doing business in the context of software-based systems.

Alternately, agile models are considered to be best suited for projects in which a great deal of change is anticipated [Cockburn, 2001; Boehm, 2002; Highsmith, 2002]. Because of the inevitable change, creating a detailed plan may not be worthwhile; the plan will only change. Spending significant amounts of time creating and inspecting an architecture and detailed design for the whole project is similarly not advisable; it will only change, too. Rather than spending a great deal of time early in the process on planning and requirements gathering, agile software developers spend time planning and gathering requirements for small iterations throughout the entire lifecycle of the project. Agile models have been around for some time, usually in the form of iterative and prototyping process model [Basili and Turner, 1975]. Some more recent examples of agile methodologies are Extreme Programming (XP) [Beck, 2000], Scrum [Schwaber and Beedle, 2002], Crystal [Cockburn, 2001], and FDD [Palmer and Felsing, 2002].

Collaboration is an emphasized philosophy of agile methodologies. Consistently, the agile methods profess that software developers should sit in close proximity to their teammates, should collaborate on all tasks, and should talk and make agreements face-to-face rather than through signed-off documents [Cockburn, 2001]. XP also strongly emphasizes the need for software development teams to work a 40-hour work week, or at a sustainable pace [Beck, 2000]. One purpose of the 40-hour work week is to minimize the impact of work on the personal life of the team members, often a concern of female IT workers [Freeman and Aspray, 1999]. Another more technical reason for the practice is that tired programmers make more mistakes that may take longer to find and fix than the overtime worked. The work arrangements of agile teams are quite different from the stereotypical insular work arrangements of solo programmers working in cubicles all day (and night), listening only to the music on their headsets. Changes in the organization of work often have consequences for recruiting and retaining employees. The transition from solo programming to more collaborative work styles of programming may make a career in IT more attractive to women and for other reasons to minorities and men who are more productive in team environments, alleviating concerns of the lack of social interaction stereotypical of most IT positions.

2.3 Gender Studies

Over the past 30 years, a considerable number of gender research studies have been conducted in the current problem of attracting and retaining women in IT careers. Kerr [1994], in writing about gifted women and girls, notes that women continually lower their career expectations beginning in adolescence and into college. Women have less interest than men in careers in engineering and the hard sciences and more interest in the social sciences. As career aspirations are lowered, a number of studies, e.g. [Arnold, 1994], find that women's confidence continues to slide downward in college, as they perceive themselves to be less intelligent than when they were in high school. College women are conflicted by pressure from peers to obtain a relationship with a "high-status" partner, and as Holland and Eisenhart [1990] note, college provides many women with an "education in romance." For these smart girls, the career aspirations of their partners become more important than their own career development. College women who do retain high expectations for themselves have high degrees of "identification" or interests in a particular career [Holland and Eisenhart, 1990]. Other factors positively affecting women's confidence are mentoring and leadership opportunities while attending college.

Researchers have studied the low interest of females in IT careers in an attempt to uncover reasons for and solutions to the problem of under-representation. The under-representation of women in the mathematical sciences emerged as a problematic issue in the 1970s while the under-representation of women in IT careers is a relatively new phenomenon. Much can be learned from the struggles to increase the participation of women in mathematics [Hanna, 1996]. One of the most significant effects was obtained from changing students, parents, and teachers' expectations and perceptions of the importance of the four years of high school mathematics. The gender gap in mathematics achievement is disappearing due to new perceptions and university entrance requirements. In many settings an equal number of young women and young men participate in high school calculus courses and major in university mathematics. Unfortunately, high school computer science is not a requirement for college entrance at most universities. In their study of high achieving girls Vouk, Berenson et al. [2004] found that the girls had little knowledge of high school computer science courses because they did know that these courses were not required for college entry and generally not weighted in terms of grading. The girls' perceptions were that computer science courses were not as "valuable" as advanced courses and would lower their grade point average, and consequently their high school ranking.

2.4 Frameworks and Questions

The guiding conceptual frameworks for our study stem from a perspective of situated cognition [Lave and Wenger, 1991] where students learn by participating in a collaborative apprenticeship, and the communication dimensions of social constructivism consisting of the univocal and dialogic functions of discourse [Wertsch and Toma 1995]. Univocal thinking is limited to the conversations that individuals have with themselves, while dialogic communications are those that occur within their interactions with others. While both the Lave/Wenger and the Wertsch/Toma frameworks speak to the social constructs of interactions, situated cognition is focused on the apprenticeship nature of the learning. In this software development course, students move from the peripheral apprenticeship positions of their first two years of study toward assuming more realistic tasks of an IT worker. In our study, we examine student' perceptions of their learning in a software development course as they move beyond their initial apprenticeships, acquiring and taking on more responsibilities of a job as a computer scientist. These contexts are the focus of study along with the univocal and dialogic nature of the learning context. We acknowledge that both univocal and dialogic communications occur within the collaborative contexts of this course.

Our research questions pertain to the collaborative learning environments created through agile software development and pair programming methods taught in an upper level computer science course. As such, we examine the following questions.

1. How can we characterize the collaborative learning environments?
2. In what ways did the collaborative learning environments appear to foster women's interests in computer science careers?

3 Method of Inquiry

In this section, we discuss the details of the course to explain the educative experiences of the students in this software engineering course. Additionally we describe the research design, the participants, the sources of data, and the analysis.

3.1 Course Details and Educational Approach

In the fall 2003 semester, 106 students were enrolled in the software engineering course at NCSU. The students were divided into six lab sections with a maximum of 24 students in each lab section. Within each week, students had two hours of lecture time and two hours of closed lab. There were three two- to three-week programming assignments. Students were instructed to work in pairs for the first two assignments, with different assigned partners for each assignment. For the third assignment, each student worked alone. The purpose of the solo assignment was to assess the students' perceptions of working alone again after working collaboratively for the prior five weeks.

The course ended with a three- to five-person, six-week team project. Within this extended timeframe, students were required to develop the system in three short iterations using the XP methodology. The project involved the creation of a software reliability estimation tool. They also had the opportunity to earn extra credit on project enhancements. For the purposes of the research, some groups were designated to be solo groups while others were designated to be collaborative groups. The intention was that the solo groups would use the "divide-and-conquer" approach in which they would each be assigned a part of the project to work on alone and would integrate their work upon completion of work tasks. The collaborative teams would work in pairs. The students were asked to express a preference to be in a paired group or a solo group. The students were assigned to a paired or solo group according to the following:

1. Students reported whom in their lab section they did not want to work with and was not assigned to work with one of these students.
2. No female worked alone in a group of other males.
3. Balancing skill level so no team had a blatant advantage (e.g. having all A students) and the paired and solo groups were as academically equivalent as possible. The measures used to determine academic advantage were midterm scores and their overall and computer science grade point averages (GPA) when the students gave us permission via Informed Consent to view their GPA scores.

The students worked with different people for the paired assignments. This dynamic co-worker assignment is often called pair rotation. A research study at NCSU [Srikanth, Williams et al., 2004] indicated that students prefer pair rotation because they enjoy working with new people and getting to know their classmates. In a five-year study of 23 coeducational computer science departments in Virginia, face-to-face interviews revealed that both men and women explained that their classmates were a necessary source of help [Cohoon, 2001]. However, the "work alone" nature of many computer science courses is not conducive to students meeting each other. In the fall of 2002, one quiet female student indicated to the third author that she wished she had pair programmed all through her university years. She said she had been taking computer science courses for three years and did not know anyone in her classes. She indicated that she now felt she knew most of the people in her lab section and that she was much happier.

3.2 Case Study

To examine the questions outlined in Section 2.4, we focused our research on three female computer science students. We used a collective case study approach to analyze emergent themes of perspectives from three women who were enrolled in the same section of a junior-/senior-level software engineering course. Collective case studies are based on analysis of multiple individuals within a specific domain and the dynamics that result from within that domain [Creswell, 1998; Huberman and Miles, 2002]. Here the domain is “collaboration” or working with others while the dynamics emanate from instructional and programming approaches used by the students.

3.3 Participants

The participants were three female students who were enrolled in an upper-level undergraduate computer science course in software engineering during the fall semester of the 2003-2004 academic year. Two of the participants, Sylvia and Beth (pseudonyms), had no prior work experience in the IT field. The third participant, Stacy, had prior work experience with a large networking company. In terms of grades, Beth was an average student, whereas Stacy and Sylvia achieved well above the class average.

3.4 Data Sources

The primary sources of data were semi-structured interviews that were audio-taped and transcribed. A professor and a graduate student, both in the College of Education, conducted the interviews of seven students to ensure anonymity of the students from the course teaching staff. The seven interviewees, three of whom were the females, were asked to respond to several questions relating to their experiences with pair programming, their reactions to the first three assignments, their career aspirations, and their experiences in the computer science field. These interviews occurred near the middle of the semester, after students had completed the first three assignments, and were beginning the team project. The interview protocol is in Appendix A.

In addition, all students in the class were asked to speak to their experiences with their team assignments by writing a two-page project retrospective. The participants wrote the project retrospectives at the end of the semester after the completion of their team project. Thus, some data were obtained conversationally with relatively little time to think about one's answers while the retrospective data were in written form with several weeks to prepare. The specific questions that needed to be answered in the project retrospective are provided in Appendix B. Some of the questions were specifically related to technical aspects of the project. For the team project, Beth was assigned to a solo team even though she requested a pair team. Stacy requested and was placed on a solo team, and Sylvia participated, as requested, in a pair team.

3.5 Analysis

The case of collaboration was analyzed by examining and then categorizing the data. Four categories emerged: a) working with others, b) productivity, c) confidence, and d) interest in software development and IT careers. Then data were categorically sorted and reexamined to reveal rich descriptions and themes of the collaborative experiences. These themes provide the conjectures found in Section 5.

4 Results

The results are presented within four major themes in these women's voices. Within the collaborative experience, the importance of face-to-face meetings emerged as a major factor of student satisfaction with the assignments and team project. Stacy, Sylvia, and Beth all agreed that within the team projects, whether solo or paired, frequent meetings were important to the quality of their products and levels of satisfaction with the collaborative experiences. The second collaboration theme to emerge was the increase in student productivity. Students agreed that when collaboration took place there was a higher quality product that took less time than a solo approach and that their time management skills were enhanced. The third theme to be identified was that of building confidence. There was general agreement among these three women that the collaborative work increased their confidence. Interest in IT careers, another major theme, was noted among all three women. The following data support these findings.

4.1 Face-to-Face Collaboration

The descriptions of the subjects' collaborative experiences of agile software development methods and pair programming highlighted the importance of meeting your partner or team members in a physical sense rather than through other forms of communications, such as e-mails or telephone calls. Beth reflected on her experiences with pair programming:

If you want to get full benefit from pair programming you really have to get together. Two persons sit beside each other doing stuff together as opposed to, you know, just meeting up one time and deciding to break [up the assignment] – *I'll do this part and you do that part. ... we meet again tomorrow or the day after and then we see what we got and go on from there. I feel that approach wouldn't be as effective as say, spending most of the time sitting together doing it.*

The approach of one person typing the code (driving) and the other observing (navigating) the coding was noted positively. Sylvia describes how working with others in another class has evolved into a pair programming approach.

I know another course that I'm doing at the moment – it's actually programming in C but I'm working with two or three other people in my building also doing it. ... One of them – we didn't actually plan the whole pair programming or anything. But when we're working on it, it was very much the role of driver/navigator – that kind of thing. Like he was typing and I was kind of watching what he was doing and asking him questions. I think it can be a very natural way to work.

In her retrospective, Sylvia gave advice to others based on how her paired team worked in comparison to other teams.

Also, meet regularly with the group and discuss what's going on, as I know that some groups had big problems with integrating their code, which is a hassle that can be avoided by just meeting up more often.

Stacy substantiated Sylvia's claim concerning the importance of the physical meetings when writing in her retrospective about her experiences on a solo team.

If next year's class does solo groups as well, I would advise them to have one meeting time at the beginning of the week's iteration and one in the middle of the week. We only had one at the beginning and then when people weren't doing their part, they just wouldn't respond to e-mail. I think they would've been more on task if we were meeting face-to-face mid-week.

For these women, whether they selected solo or paired teams, there was general agreement that e-mail or telephone communications did not replace the face-to-face meetings. Whether it was to keep the lines of communications open, increase on-task behaviors among team members, or improve the understanding of the development and coding; the benefits of physical, face-to-face meetings were apparent.

4.2 Experiencing Productivity

Less Time. A consensus developed among the students that paired programming and collaborative development methods took less time than solo work. Since students value time as an important factor in their lives, their opinions concerning structured collaborative experiences were positive. In speaking about her experiences with her partner, Sylvia commented:

We just sat down and programmed and it was just the way we worked. It worked out really well. ... We kind of got through this stuff in so much faster time than we would have separately – Much, much faster. ... if you don't know how to do something and you're going away to search something on the Internet or look it up yourself, it's easier to have someone else explain it to you.

In speaking of her solo assignment, Beth speaks to feeling somewhat lost without a partner and taking a lot of time just staring at the computer.

Then you go back to the old style of doing solo, it's kind of ... a little bit hard. It suddenly feels like the weight is heavier – like you have to do all this stuff on your own and there's nobody to talk to and to ask a question to. So you have to like stare at the computer for hours ... thinking all by yourself.

Speaking in a similar vein, Stacy infers that working alone can increase the amount of time she may have to spend on an assignment.

I think that it's very common thing to get sort of stumped on something or code something that you just look back at and can't figure out why it's not working. I think that pair programming really helps solve those issues, 'cause you have one other person there looking with you and they can quickly see what you are blind to for whatever reason.

Time Management. The issue of collaborative work taking less time was repeated throughout the transcripts and the retrospectives. Also, students commented that they produced a higher quality product when they work collaboratively in pairs or teams. This theme is embedded within discussions of time management. More than 75% of the students in the class, including the three case subjects, did not finish the solo assignment due to issues of time management, thereby, reducing the quality of their assignments. In her interview, Sylvia explained her incomplete solo assignment as follows.

That didn't go well, But I didn't … with Fall Break and everything I went on [vacation]. *So I didn't get started on it until kind'a the day before which … I mean I knew I wasn't going to get it done, but I had so much else to do. I didn't, you know,… I just wasn't prepared.*

Time management affected the quality of Beth's first pair programming assignment.

Well actually I think I ran into a problem. We didn't complete the assignment… and our output [was] *not completely correct. … We ended up sitting in the lab and it's due at 11:45 … we were doing the testing right up to that time... so it feels* [like] *a lot of pressure because we didn't meet much in the first week. That's the main thing.*

On the other hand, she speaks about the high quality of her team products based on their frequent and timely face-to-face planning meetings.

We're going at a [good] *pace, I feel. If we continue to do that I think we'll be able to finish the stuff a little bit earlier than the due date so we will be able to attempt the … extra credit.*

Throughout the interview, all three students expressed feelings of frustration and discomfort when they experienced poor time management on their part or the part of team members. High quality products were associated with timely meetings of the pair or team.

4.3 Building Confidence

Within the focus of the collaborative pair programming and team planning, these three women became more confident over the 15 weeks of the course. Each began their first pair programming assignment with a negative experience. Sylvia's first partner went off and completed their pair assignment on his own.

… It was very strange having to work with someone else and my first assignment didn't go very well in that he went away and did the whole thing and didn't ask me anything about it … I was very annoyed … I had all the ideas in my head and we were supposed to meet up and start coding and he came back and said, "I got it done." But I think it turned out afterwards he probably should have [met with me] *'cause we had one or two errors in it that I would have spotted straight away.*

As previously noted, Beth and her first partner had time management issues. She explained their procrastination further in the interview.

I think like at the beginning we just didn't realize timing issues that are critical because it's a first time for both of us [using] *pair programming. ... we sort of tended more like when we do the solo. You know, both of us don't have time to meet today. We will meet tomorrow, or push it back to the day after.*

Stacy's initial experience was stressful because her partner dropped the class two days before the assignment was due.

She said, "I'm sorry we can't even get together because I'm going to be on vacation and so we're just going to have to work separate and then put our parts together." So, she divided up our work and then the day we were supposed to come back and put them together – like two days before it was due – and when I got in the lab – [she said] "I didn't do any of my part when I was on vacation so I'm dropping." So that wasn't the best. That gave me a little bit of a not so good view of the pair programming just because my partner ditched on me, but the second time the pair programming worked better.

Yet after these initial experiences, which may be essential to learning to work together, these women expressed positive feelings about their productivity within the collaborative experience. Stacy revealed her feeling of relief and confidence when she has a programming partner.

I'm always running into problems where I get stumped and when I try to debug my own code, I have the hardest time figuring out why what I did is not perfect. So it's a lot easier to have somebody there. I ... took a break from school and did some co-op and stuff and so it's actually been about a year before this class before I had done Java programming and I was not so fresh on it. So the pair programming really helped me because I would sit there [with my partner] *and I would go, "Now there's a method and it's something you can token on strings." Then they would go, "Oh, yeah, there's a string tokenizer," and I'd just type it in – you know it was a lot easier. I didn't have to go just searching through the APIs for hours.*

Sylvia details why face-to-face collaborative ways of working helped her to avoid errors in her reasoning. Less time and a higher quality product appear to build her confidence.

It's definitely good for coding as well. You know, if you're writing something and the other person is like "How is that going to work?" You would explain the whole thing to them. ... If you're explaining your reasoning you see flaws easier, you see flaws in your reasoning whereas if you do it on your own you're probably going to go away and code the whole thing and then suddenly you realize ... ooops ... "I don't know what I was actually doing here."

As seen in previous comments, Beth expressed feelings of not being confident in her abilities. However, she did reveal that she received affirmation from her partners who expressed their weaknesses in understanding. These exchanges seemed to provide a sense of relief to Beth, realizing that she was not the only one who does not know "the answer." When asked by the interviewer if it was helpful communicating and talking with the other person, Beth responded:

Yeah, definitely. Because in a situation where you don't understand some stuff, you can ask the other person. The other person maybe says, "Yeah, I understand it," or "No, I don't understand it either." ... Then you know you're not the only person – somebody else is having the same problem. ... If you're on your own and you have some problem – you don't understand - you sort of tend not to ask people as opposed to two of you – none of you understand it. ... Then you don't feel any embarrassment. ... You tend to be more open with the other person, where if you're working well together then you tend to be really open. You can learn a lot from that.

The partners take turns being the teacher and the taught, from moment to moment. For example, this female student could more easily overcome the potential feeling of having inferior knowledge to a male because she, in turn, helped another student.

For instance, HW2 - I felt that [male] was more knowledgeable that I in white box testing so he took the time to explain things to me. Then with HW6, [female] was lost, but I actually explained the concept of JSP and HTML to her after I learned it from the TA. It's just a big learning circle where no one feels intimidated because we all are peers . . . I'm doing better than I would if I had to work alone.

For the stronger students, the collaborative experience served to enrich the learning environment. Having a partner there to fill in the gaps in her knowledge increased Stacy's confidence in her Java programming abilities. In a somewhat different vein, Sylvia found that a good partner could slow down her thinking, catching errors in her reasoning, before she became lost in a plan that was not productive. The average student received assurances from her partners who may not have understood parts of the assignment either. Beth noted that she felt less embarrassment and was more open with her partners than her classmates, all of which contributed to her increased confidence.

4.4 Increasing Interest in IT Careers

Each woman spoke with enthusiasm of her future career in IT. While Sylvia and Beth were aiming toward careers in software development, Stacy saw herself participating in a networking career, an extension of her internship.

Well, it's kind of weird because you do all the configurations on the network – it's almost like it's own sort of programming like you have to know the language to type in all the codes and stuff. So I kind of felt like I got a little bit of the programming that I like, but it also combined with hardware. So I felt like it was kind of a neat combination 'cause I like hardware, too.

Sylvia shares her career interests saying, *I want to go into software development – the actual coding and all that kind of thing is where I'd be most interested.* Later she expresses confidence about her vision of the future workplace. *...it's probably the same for everyone, that they don't really know what they're going to get themselves into, so I'm not too worried that way*.

In a similar vein, Beth shared her ideas about a future IT career.

Well, I hope I end up in software development where that's my main, major area. Mostly we'd probably start with software testing. I'm doing one course in that as well, but I hope I really end up developing stuff, developing tools or whatever.

The voices of these women have spoken to the value they placed on working together using pair programming and agile software development. They noted the importance of time management in increasing the quality of their completed assignments, speaking with confidence about their experiences. All three anticipate completing their degrees to work in IT careers.

5 Conjectures and Implications

The type of qualitative inquiry reported in this paper does not produce generalizeable results. The case study is a tool for advancing ideas or conjectures to be tested with other subjects at another study. When writing the conjectures, there are three parts that we address: the conjecture statement; an explanation of the statement; and a discussion of related research results available in the literature. Larger quantitative studies can then be designed to evaluate the generalizability of the case study's conjectures. In this section, we propose three conjectures:

Conjecture 1. Effective collaboration is not accomplished via piece-meal, "divide-and-conquer" project management.

Frequent, face-to-face meetings appear to be a requirement for timely, high quality products. However, several students in this research had bad initial collaboration experiences from which they were able to recover within the next pair programming assignment. Waite et al. [2004] suggest several strategies to improve students' collaborative skills, which may alleviate this problem. Their suggestions include fostering classroom discussion, use of group decision making exercises, and emphasizing the instructional value of the assignments instead of focusing on the end products. Johnson, Johnson et al. [1991] claim that successful collaboration of undergraduate college students involves face-to-face "promotive interaction" where students help each other learn from sharing a mutual goal.

Conjecture 2. Collaboration helps female students build confidence with higher quality products.

Margolis and Fisher [2002] found that once-enthusiastic female college students find themselves in a descending spiral of interest through the corroding effects of lack of confidence, negative comparisons to peers, poor pedagogy, and biased environments. Conversely, when students perceive value and quality in their products, there is an increase in their perceptions of their abilities. The confidence levels of the women in this study grew over time as they gained experience from working in collaborative situations. Using the resources of a partner or planning together more effectively to complete the assignment on time contributed to their sense of accomplishment. Research has shown that women who are confident in their activities will retain interest in those activities. Lancaster and Smith [1994] found the use of cooperative educational practices helps to diminish the influences that cause women to lose self-confidence, particularly in traditionally male-dominated fields such as computer science. Moreover, Johnson, Johnson et al. [1991] assert that students working together have more confidence in the value of their ideas. In collaborative situations, it is important for all participants to feel their ideas are worth sharing. By increasing the confidence levels of these women who, in turn, share their ideas, higher quality products are produced by the team in addition to the positive effects on the women, themselves.

Conjecture 3. Effective collaboration may help students manage time more effectively.

We infer that a reduction in the time of the task with a corresponding increase in the quality of completed task can result from structured collaborative experiences that require face-to face-meetings. Nespor [1994] investigated the time-intensive requirements needed to study physics and Eisenhart and Finkel [1998] noted the time demanded of undergraduate engineering majors. Several studies [Mackavey and Levin, 1998] note that females, even as undergraduates, have more time constraints than males. The relational responsibilities of family, friends, and significant others require women to spend more time away from their studies [Mackavey and Levin, 1998].

Margolis and Fisher [2002] conducted a longitudinal study following the students' experiences for two to four years including 51 female and 46 male computer science majors at Carnegie Mellon University [Margolis, 1997]. In the interviews, the researchers found evidence of a cultural stereotype of the successful computer science student as being a person who is at the computer "twenty-four hours a day/seven days a week ('24/7'), living, thinking, and breathing computer science." [Margolis, 1997] Female students are concerned that their study of computer science may require a myopic focus of the computer, may detach them from people, and could require 24/7 dedication. Margolis and Fisher [2002] express their concern that many women students are hesitant to join the computer science world, where they fear that the links to other interests in their lives will disappear gradually. Twenty percent of the female students interviewed questioned their choice of major in computer science because they felt that they do not share the same type of focus and intensity of interest they see in their male peers. [Katira, 2004] We observe that pair programming and agile development can aid in this concern because the students are not only interacting socially with people while they do their work, but they are completing tasks faster – allowing the students to pursue other life interests.

The implications of this study are found in the relationship between collaboration and time, confidence, and interest in IT careers among women software engineering students. The data provide an image of collaboration and the dynamics that emanate from a number of collaborative experiences in a software engineering course. Connections are made beginning with the observation that an essential ingredient of the collaboration is face-to-face meetings to accomplish the successful completion of assignments. Whether coding or planning, these meetings were deemed essential for success. Students observed that they were more productive when working collaboratively, taking less time and producing a higher quality product. With more productivity, these women experienced more confidence and consequently more interest in IT careers. As shown in Figure 1, these factors build a disposition of apprenticeship within IT careers, and as Stacy commented based on her solitary IT work experiences in a networking company,

> *… from my experiences in this class I would say that I would love to be part of some kind of programming team where we were able to kind of feed off each other and work closely together rather than just being like locked off in an office by yourself for a month or something.*

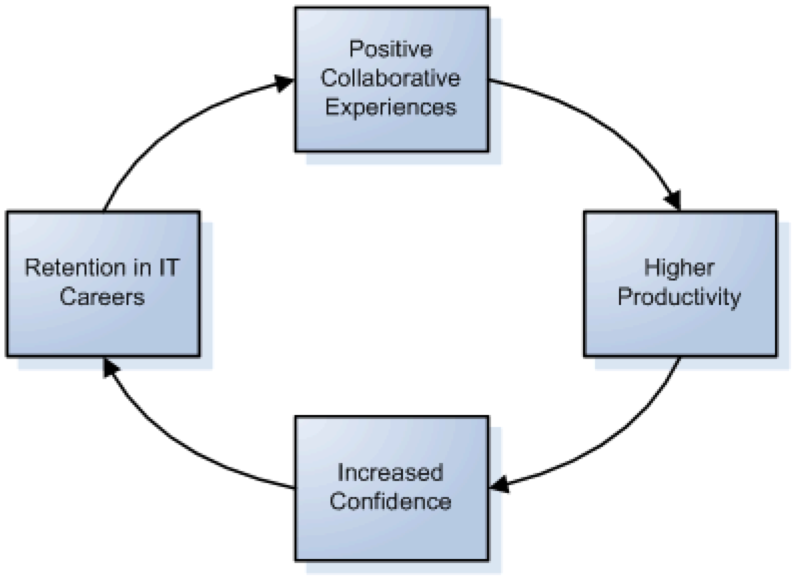

Figure 1. Effects of a Collaborative Apprenticeship on Female Students' Interests in IT Careers

6 Summary and Future Work

Our initial intent was to characterize the collaborative learning environments of pair programming and agile software development. The collective case study of three females in the class revealed four themes common to each: working with others; productivity; confidence; and interest in IT careers. Three conjectures concerning collaboration emerged from our study and included the importance of face-to-face meetings, an increased confidence among women based on product quality, and a significant reduction in the amount of time spent on assignments. Additionally, we propose a model for future testing that connects these three factors with an increased interest in IT careers.

There have been few pedagogical approaches to address the problem of time issues among undergraduates. Here we conclude that pair programming and agile software development techniques are pedagogical alternatives to improve undergraduate computer science while working to retain females in IT careers. Our future work will test the time management issues of the model proposed in Figure 1 within the collaborative contexts of pair programming and team projects. While time is a very important value for women, we propose to test the hypothesis that time is also highly valued by most men. Case studies of collaborative work environments are of additionally interest to us in terms of less time, higher quality, and more job satisfaction.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 00305917. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Arnold, K. (1994). Academically talented women in the 1980s: The Illinois Valedictorian Project. Women's Lives through Time: Educated Women in the Twentieth Century. K. Hulbert and D. Schuster. San Francisco, Jossey-Bass.

Basili, V. R. and A. J. Turner (1975). "Iterative Enhancement: A Practical Technique for Software Development." IEEE Transactions on Software Engineering **1**(4).

Beck, K. (2000). Extreme Programming Explained: Embrace Change. Reading, Mass., Addison-Wesley.

Berenson, S., K. M. Slaten, et al., "Collaboration through Agile Software Development Practices: Student Interviews and Lab Observations," North Carolina State University Department of Computer Science, Raleigh, NC TR-2004-12, April 26, 2004.

Bevan, J., L. Werner, et al. (2002). Guidelines for the User of Pair Programming in a Freshman Programming Class. Conference on Software Engineering Education and Training, Kentucky.

Boehm, B. (2002). "Get Ready for Agile Methods, with Care." IEEE Computer **35**(1): 64-69.

Cockburn, A. (2001). Agile Software Development. Reading, Massachusetts, Addison Wesley Longman.

Cockburn, A. and L. Williams (2000). The Costs and Benefits of Pair Programming. Extreme Programming and Flexible Processes in Software Engineering (XP2000), Cagliari, Sardinia, Italy, Addison-Wesley.

Cohoon, J. M. (2001). "Toward improving female retention in the computer science major." Communications of the ACM **44**(5): 108-114.

Creswell, J. (1998). Qualitative inquiry and research design: Choosing among five traditions. Thousand Oaks, CA, Sage.

Eisenhart, M. and E. Finkel (1998). Women's science. Chicago, University of Chicago.

Freeman, P. and W. Aspray, "The Supply of Information Technology Workers in the United State," Computing Research Association, Washington, DC.

Hanna, G. (1996). Towards gender equity in mathematics education. Boston, MA, Kluwer.

Highsmith, J. (2002). Agile Software Development Ecosystems. Boston, MA, Addison-Wesley.

Holland, D. and M. Eisenhart (1990). Educated in romance: Women, achievement, and college culture. Chicago, IL, University of Chicago.

Huberman, A. and M. Miles (2002). The qualitative researcher's companion. Thousand Oaks, CA, Sage Publications.

Johnson, D. W., R. T. Johnson, et al. (1991). Active learning: Cooperation in the college classroom. Edina, MN, Interaction Book Company.

Katira, N. (2004). Understanding the Compatibility of Pair Programmers. Department of Computer Science. Raleigh, NC, North Carolina State University.

Kerr, B. (1994). Smart girls. Scottsdale, AZ, Gifted Psychology.

Lave, J. and E. Wenger (1991). Situated Learning: Legitimate peripheral participation. New York, NY, Cambridge University Press.

Mackavey, M. and R. Levin (1998). Shared purpose: Working together to build strong families and high-performance companies. New York, AMACOM.

Margolis, J. and A. Fisher (2002). Unlocking the Clubhouse: Women in Computing. Cambridge, Massachusetts, The MIT Press.

Margolis, J. a. A. F. (1997). Geek Mythology and Attracting Undergraduate Women to Computer Science. Joint National Conference in Engineering Program Advocates Network and the National Association of Minority Engineering Program Administrators.

McDowell, C., L. Werner, et al. (2002). The Effect of Pair Programming on Performance in an Introductory Programming Course. ACM Special Interest Group of Computer Science Educators, Kentucky.

McDowell, C., L. Werner, et al. (2003). The Impact of Pair Programming on Student Performance of Computer Science Related Majors. International Conference on Software Engineering 2003, Portland, Oregon.

Nagappan, N., L. Williams, et al. (2003). Improving the CS1 Experience with Pair Programming. SIGCSE 2003.

Nespor, J. (1994). Knowledge in motion: Space, time and curriculum in undergraduate physics and management. Washington, DC, Falmer.

Palmer, S. R. and J. M. Felsing (2002). A Practical Guide to Feature-Driven Development, Prentice Hall PTR.

Schwaber, K. and M. Beedle (2002). Agile Software Development with SCRUM, Prentice-Hall.

Seymour, E. and N. Hewett (1997). Talking about leaving: Why undergraduates leave the sciences. Boulder, CO, Westview.

Srikanth, H., L. Williams, et al. (2004). On Pair Rotation in the Computer Science Course. Conference on Software Engineering Education and Training, Norfolk, VA.

Vouk, M., S. Berenson, et al. (2004). Women and Information Technology (WIT): A Comparative Study of Young Women from Middle Grades through High School and into College. Annual Principal Investigators Meeting, National Science Foundation, University of Pennsylvania. Philadelphia, PA.

Williams, L. and R. Kessler (2003). Pair Programming Illuminated. Reading, Massachusetts, Addison Wesley.

Williams, L., R. Kessler, et al. (2000). Strengthening the Case for Pair-Programming. IEEE Software. **17:** 19-25.

Williams, L., E. Wiebe, et al. (2002). "In Support of Pair Programming in the Introductory Computer Science Course." Computer Science Education **September**.

Williams, L., K. Yang, et al. (2002). Pair Programming in an Introductory Computer Science Course: Initial Results and Recommendations. OOPSLA Educator's Symposium, Seattle, WA.

Williams, L. A. (2000). The Collaborative Software Process PhD Dissertation. Department of Computer Science. Salt Lake City, UT, University of Utah.

Appendix A: Interview Protocol

October 2003

OPENING:

SAY: We are evaluating the instructional approaches used in your computer science 326 course. It is important for the instructors to know how well these new methods are working for you. This is why we want to ask some of the students what they think about the programming assignments in this course.

Your responses will be anonymous. While we will share the results of the interviews with the instructor, he or she will not know that you were interviewed.

1) What do you think of the assignments so far?
   - a) Can you explain why you think that?
   - b) Can you give me an example?

2) So some of the assignments have been paired and some have been solo. What do you think the students in your lab prefer, pair or solo?
   - a) What reasons do they give for preferring _____________?
   - b) What about those who prefer _____________?

3) What about you? What approach do you prefer?
   - a) Why is that true?
   - b) Any other reasons?

4) What makes _________ an effective instructional tool for you?
   - a) Have you noticed other approaches that help you learn?
   - b) Can you give me an example?

5) What do you see yourself doing after graduation?
   - a) Have you ever done this before? How did it go?
   - b) How do you envision the workplace?

6) Do you think pair programming will work in today's IT workplace?

a) Why? Please explain this idea some more.

b) Why not? Please explain this idea some more.

Thanks very much for your time. You were very thoughtful. Your ideas will help many students here at NC State and in other programs across the United States. Good luck with the semester.

Appendix B: Project Retrospective Protocol

Turn in a two- to three-page document that enumerates honestly and constructively what worked and didn't work well with your team project. At a minimum, specifically discuss:

1. User Stories, Iteration, Acceptance Tests: Did having three short iterations help you pace yourself to completion? Do you feel you benefited from feedback you got after each iteration? Did having the Acceptance Tests help you to clarify the requirements?
2. Configuration management: Did CVS help you manage your files?
3. JUnit and FIT: Do you feel either or both of these kinds of testing helped you in creating a high quality project? Do you feel like this kind of testing helped reduce debugging time? Did you create these test cases as you went along or did you complete a user story and then run the tests? Based on your experience, do you think it is best to create the tests all along or at the end?
4. Your method of work: Pair programming or Solo programming. How did it work? What percentage of time do you feel like you worked solo, what percentage of the time do you feel you worked in pairs? What made you decide when to work solo and when to work in pairs [time constraints, difficulty of work, etc.]
5. How well your team communicated with each other. What vehicles did you use for communication – web site, email, setting up a mailing list, etc.
6. Division of team roles – both the team roles discussed in class (team leader, development manager, quality manager. planning/process manager) . . . but also did you assign a certain person to be the SWT expert, etc. How was technical information shared within the team?
7. Did you enjoy the project? Why or why not? Did you like the fact that it was an Eclipse plug-in? [This question will help immensely for choosing next year's project.]
8. Did you reuse any code found on the web? How did this go? How was testing this code?
9. How did you like Extreme Programming? Do you think your project would have instead been better run in a plan-driven way? Did you formally or informally create use cases, class diagrams, or sequence diagrams? If so, did you share them among the team? Did you run any CRC card sessions?
10. What advice do you have for next year's class that will have a 4-5 person, 6 week team project [which may or may not be an Eclipse plug-in?]

11. Did you have any particularly stubborn defects? Can you analyze the cause of these defects – is there anything you'd change about what you did to prevent such defects?
12. Analyze the effect of keeping Severity 1 and Severity 2 defects out of your project. Do you think you ended up with a better structured code base because of them?
13. Please include anything else you'd like to share.