Eleven Guidelines for Implementing Pair Programming in the Classroom

Laurie Williams[1], D. Scott McCrickard[2], Lucas Layman[1], Khaled Hussein[2]
[1]North Carolina State University, Department of Computer Science
{lawilli3, lmlayma2}@ncsu.edu
[2]Virginia Polytechnic Institute and State University, Department of Computer Science
{mccricks, khussein}@cs.vt.edu

Abstract

Utilizing pair programming in the classroom requires specific classroom management techniques. We have created nine guidelines for successfully implementing pair programming in the classroom. These guidelines are based on pair programming experiences spanning seven years and over one thousand students at North Carolina State University. In Fall 2007, pair programming was adopted in the undergraduate human-computer interaction (HCI) course at Virginia Tech. We present the pair programming guidelines in the context of the HCI course, discuss how the guidelines were implemented, and evaluate the general applicability and sufficiency of the guidelines. We find that eight of the nine guidelines were applicable to the Virginia Tech experience. We amended our peer evaluation guideline to account for constantly supervised pairing, as was the case at Virginia Tech. We add two guidelines stating that a pair should always be working toward a common goal and that pairs should be encouraged to find their own answers to increase their independence and self-confidence.

1. Introduction

Research suggests that pair programming has many pedagogical benefits. Pair programming creates an environment conducive to active learning and collaboration, helps to lower student frustration with challenging problems, and increases programming self-confidence and interest in information technology [1]. To leverage the benefits of pair programming, educators must create an effective pair programming environment in the classroom. Implementing effective pair programming requires several specific classroom management techniques.

Successful pair programming requires discipline on the part of the students and positive reinforcement on the part of the teaching staff. Based upon our experiences, we previously documented classroom management guidelines in hopes of enabling other educators to be as successful as possible with pair programming [13]. The first and third authors from North Carolina State University (NCSU) have used pair programming extensively in CS1, undergraduate software engineering, and several graduate-level courses over the last seven years, involving more than one thousand students. Through this experience, many lessons have been learned and policies and practices have been adapted to be more successful with the collaborative pedagogy of pair programming.

In this paper, we describe nine guidelines for successfully implementing pair programming in a classroom or lab environment. During the Fall 2007 semester, these guidelines were followed to varying degrees as pair programming was adopted in the upper-level undergraduate course in human-computer interaction (HCI) at Virginia Tech. The objective of this paper *is to evaluate and evolve our guidelines by retrospectively comparing them with the classroom management practices and qualitative experiences at Virginia Tech.*

The rest of this paper is organized as follows: In Section 2 we provide related research results in pair programming. In Section 3, we provide information about the HCI course at Virginia Tech. In Section 4, we present the NCSU pair programming guidelines and provide observations regarding the extent to which each of these were used at Virginia Tech. In Section 5, we evolve our guidelines. We summarize in Section 6.

2. The Positive and Negatives Aspects of Student Pair Programming

Much of the research on pair programming in an academic environment has concentrated on evaluating the efficacy of the practice. Studies conducted at NCSU [2, 7-9, 12] have shown that pair programming creates an environment conducive to more advanced, active learning and social interaction, leading to students being less frustrated, more

978-0-7695-3321-6/08 $25.00 © 2008 IEEE
DOI 10.1109/Agile.2008.12

confident, and more interested in IT. The benefits to pair programming contrast with the negative aspects of traditional solo programming pedagogies, which can leave students feeling isolated, frustrated, and unsure of their abilities. These benefits appear to help increase retention in computer science, particularly among women [15].

Pair programming encourages students to interact with peers in their classes and laboratories, thereby creating a more communal and supportive environment. Students of the current Millennial generation place particular value on collaborative environments [10]. Furthermore, the collaboration inherent in pair programming exposes and reinforces students to the collaboration, teamwork, and communication skills required in industry. A study of undergraduate students at Pace University found a positive correlation between out-of-class collaboration and student achievement based on student projects and examination grades [5].

Pair programming also benefits the teaching staff. Less grading is required due to half the number of assignment submissions. A pair of students can oftentimes figure out the low-level technical or procedural questions that typically burden the teaching assistants in the laboratory [4, 16], instructor's office hours and email inbox. Finally, there are fewer "problem students" to deal with because the peer pressure involved in pair programming encourages all students to be active participants in the class. Students become concerned about jeopardizing their partner's grade and work harder on assignments, often getting started earlier than if they worked alone (though not all students report starting earlier [11]).

Alas, there are some costs to implementing pair programming. Most of the costs for educators are outlined in the guidelines of Section 4. For students, there are two major costs that persist without apparent recourse. First, we observe that a small segment of students (approximately 5%) will always desire to work alone. Most often, these are the top students who do not want to be "slowed down" by another student and who do not see benefit in teaching others. Another problem for students is the need to coordinate schedules when pair programming is required outside of a classroom or laboratory setting.

3. Pair Programming at Virginia Tech

Twenty-two students were enrolled in the HCI course offered in Fall 2007. The second and fourth authors of this paper were the course instructor and teaching assistant (TA), respectively. The course met twice weekly for 75 minutes. Prior to taking the HCI course, the students had taken three object-oriented programming courses (C++ or Java), and the course prerequisite was a second-semester data structures course. Students did not use the pair programming practice in these prior classes, though two students in the class had pair programming experience from summer internships.

The HCI course consisted of weekly lectures, generally paired with an in-class activity on the second class session of each week. However, during weeks 9 and 10 of the 15-week semester, both the lectures and activities were replaced by four pair programming sessions. During these sessions, students implemented a navigation interface for Tablet PCs using the C# 2.0 language.

The pair programming assignment was the third phase of a four-phase group project effort in which the students gathered requirements, designed, implemented, and tested navigation interfaces for on-campus security officers. The students worked in groups of three or four during the requirements, design, and testing phases. The implementation phase was conducted as a series of pair programming efforts, with each individual responsible for a unique part of the implementation of their group's project. The project was worth 40% of each student's overall grade, with the pair programming phase worth 25% of the project grade (and thus 10% of the overall course grade). The pair programming session replaced an identically-weighted group prototyping assignment from previous semesters that did not require them to program fully-functional prototypes. In these past semesters, students instead used tools like PowerPoint to create semi-functional prototypes.

Traditionally, there is no formal lab time for the course, but for this instantiation of the course four class sessions were converted to lab-style pair programming sessions. The sessions took place in a classroom with single-person desks; no lab room was available where teams could meet at tables as is recommended for pair programming. The students arranged their desks such that pairs were seated next to each other, each with a personal laptop. The student pairs spread throughout the room to isolate themselves from other pairs.

The students began the implementation process in class in the pairs and were required to complete it outside of class (alone or in pairs), if necessary. The pairs were formed using the PairEval[1] system (discussed in Section 4.4). The pairs were assigned for the first and section session and then reassigned for the third and fourth sessions. Each student in the

[1] PairEval can be freely downloaded for use at http://agile.csc.ncsu.edu/pairlearning/paireval.php.

pair served as driver and as navigator two times in the four sessions.

The pair programming phase required each individual in the class to turn in a unique program, with the understanding that they would be expected to complete the phase using pair programming. The phase required each student to code a unique information display and interaction method for the project. For example, one student might create a pie-chart menu system and another might create a map-based system and another a series of popup boxes. In so doing, the assignment resulted in different display and interaction options for accessing the interface, which were then tested by the project groups in the usability evaluation phase.

4. Guidelines

In this section, we provide guidelines [13] and rationale for implementing pair programming in the classroom based upon our experiences at NCSU and reflect on whether the class at Virginia Tech utilized these guidelines. Our guidelines are similar to but expand guidelines proposed in 2002 [3].

4.1. Supervised pairing experience

Guideline 1. *Students need training in pair programming in a supervised setting to experience the mechanics of successful pairing.*

The instructor cannot assume that the students will know what to do if they are told to pair program. The students may feel the idea is to divide the work into two parts, each student doing half. The students should be made aware that they are to work together at one computer in driver and navigator roles, they need to switch roles, they both need to be active participants at all times, and so forth. Resources for educating students about pair programming, including a 15-minute video, can be found at http://agile.csc.ncsu.edu/pairlearning/.

The watchful eye of a trained instructor or TA is essential for making sure that students are properly assuming the roles of driver and navigator, switching roles periodically, and that both students are engaged. A driver who never navigates may dominate the session, and a navigator who never drives may become disinterested. Furthermore, students benefit from "bonding" with their partner by working on a joint project in a structured setting and will be more comfortable meeting outside of class. We strongly advise against pairing first or second year undergraduate students if no pair programming will occur in a closed lab or classroom setting that is monitored by a member of the teaching staff.

Observations. Prior to starting the pair programming sessions, the Virginia Tech students had a lecture that introduced them to the key concepts of pair programming. A few students indicated that they had previously used pair programming as part of summer internships and jobs. This small number of students viewed pair programming negatively and as a burden to their workload without seeing much benefit. Care was taken to respond to the criticism calmly and rationally to avoid having these students "poison" the experience for others.

After an initial information exchange period (around five minutes) when the pairs would get to know each other, they settled down to work on the assignment. They were told that any part of the assignment not completed in class would have to be finished on their own, providing motivation to make significant progress in the class sessions. The pairs were told to remain in their assigned roles (driver or navigator), and they seemed to do so without complaint. It was unclear how much of the navigator effort was dedicated to helping the driver and how much was spent on other tasks like email or web surfing. However, there were no complaints about this from any of the drivers.

Students made unexpectedly good progress in the first session. The navigators for most teams were able to find code libraries unknown to the instructors and not referenced by the assignment. These resources enabled faster completion of the assignment. Later sessions seemed to lack the burst of enthusiasm noted in the first session. The first session was by far the loudest, perhaps because of the excitement (and to some degree the confusion) of a new development technique. However, in later sessions the navigator seemed to better understand the role to be undertaken.

4.2. Teaching staff pair management

Guideline 2. *Teaching staff must actively engage in the management of pair interactions.*

In solo programming labs, TAs spend most of their time answering questions of the students. In a paired lab, the role of the TA changes to some degree from technical assistant to proactive monitor. In paired labs, the number of technical questions put to the TAs is lower than in solo labs because pairs can usually figure out most aspects of an assignment together. Questions from pairs tend to be focused more on learning objectives and concepts rather than technical hang-ups, and may require more time per question (though this is time well-spent). With the remaining time, the TA needs to proactively visit the pairs, ask how they are doing, and ensure that they are working together effectively. The TA must look for dysfunctional pairs who are not working well together and take proactive steps to remedy the

problem. The TA must also ensure that the students switch roles periodically.

Observations. During the first pairing, it was obvious that most students were paired with partners that they did not know well. For the first few minutes, they exchanged general information about programming knowledge and interface building expertise. After about five minutes, the students settled into the assigned work with no prompting from the TA.

The TA was present throughout the pair programming sessions. He walked through the room, observing the pairs at work and answering any questions that arose. When appropriate, he would redirect questions to the pairs in hopes of helping them identify ways to find the answers themselves.

4.3. Attendance and tardy policy

Guideline 3. *Strict attendance and tardy policies are necessary to protect students from a non-participatory partner.*

A student who does not attend lecture impacts his or her ability to work effectively on and to contribute fairly to a paired project. A strict attendance policy enables the teaching staff to identify early those students who are no longer participating in the class but have not dropped it.

A persistently tardy pair programming partner puts his or her partner at the disadvantage in not having a collaborator. We recommend that a student is reassigned to a different pair if their partner does not arrive before a specified period of time (e.g. 10 minutes). A student who arrives late must work alone with a penalty on the lab assignment. The penalty for being tardy is important or students who would rather work alone will "conveniently" show up late so that he or she can work alone.

Observations. The professor and teaching assistant made it clear during multiple class sessions early in the semester that absence or lateness to the pair programming sessions would result in a grade of zero on that portion of the assignment. This policy seemed to be well understood by the students, as there was not a lateness/absence problem for the pair programming sessions despite several students who were chronically late or absent at other times during the semester.

4.4. Peer evaluation

Guideline 4. *Instructors should provide a systematic mechanism for obtaining students' feedback about their partners and must act upon the feedback when indications are a student is not being an equal participant.*

Some students attempt to escape an assignment without doing the work, thereby shortcutting the lessons the assignment is designed to teach. By having students systematically evaluate their partner's performance, students are motivated to participate equally in pair programming. Furthermore, the process helps to identify non-participatory students.

We have created an online peer evaluation system known as PairEval. Based upon the peer rating system by Kaufman et al. [6], the students are asked to choose one of the nine key words in the bullets below (short descriptions are provided to the students as well) to describe the contribution of his or her partner. The students choose among the following ratings:

1. **Excellent.** Consistently went above and beyond—tutored teammates, carried more than his/her fair share of the load
2. **Very good.** Consistently did what he/she was supposed to do, very well prepared and cooperative
3. **Satisfactory.** Usually did what he/she was supposed to do, acceptably prepared and cooperative
4. **Ordinary.** Often did what he/she was supposed to do, minimally prepared and cooperative
5. **Marginal.** Sometimes failed to show up or complete assignments, rarely prepared
6. **Deficient.** Often failed to show up or complete assignments, rarely prepared
7. **Unsatisfactory.** Consistently failed to show up or complete assignments, unprepared
8. **Superficial.** Practically no participation
9. **No show.** No participation at all

In choosing a word to describe the contributions of their partner, students generate a wider range of responses than when previously asked for a numerical rating. In PairEval, students can also provide text comments describing the rationale behind their rating.

PairEval generates a summary report of peer evaluations for each assignment. Instructors can see the rating for all previous assignments for every student so trends in participation can be observed. If a student gives his or her partner a low overall rating (e.g. "Marginal" or below), the PairEval summary report marks the student in red. The teaching staff can review the evaluation more carefully and contact the student if necessary.

At NCSU, our policy is that if the instructor meets with a student and determines that the student made little or no effort on a partnered assignment, the student will have his or her grade reduced and the partner's grade will be correspondingly increased. We recommend a strict policy whereby a student's score can be multiplied by their contribution (e.g. if

they did 50% of what they were supposed to, they get 50% of the score) and their partner's grade is boosted by 50%. Furthermore, part of the students' grades should be contingent on completing peer evaluations to ensure that peer evaluation occurs.

When pairing students for homework assignments, prompt attention and adequate consequences are essential to bring potential "freeloaders" back into a contributing role. After the instructor handles a few of these instances, an environment of participation is created in the classroom and instances of freeloading become rare. We recognize that not all problem students will be identified because sometimes students are reluctant to file negative reports on their partners.

Observations. At Virginia Tech, students completed a PairEval evaluation on their partner after each pair programming session. The Virginia Tech team did not have a clear sense of how to adjust grades based on paired participation. Without a clear impetus, the students were often slow in completing the evaluations. Furthermore, all paired work was done in the classroom under the watchful eye of the TA and non-participation was not a problem that needed to be acted upon by the teaching staff.

4.5. Balancing individual and collaborative work

Guideline 5. *In each course, students should be evaluated on a balance of individual and collaborative work.*

We want to ensure that individual students are learning the course material, and that some students are not relying solely on their partners. Consequently, students should be evaluated by their performance on both individual and collaborative assignments. The individual-collaborative balance may change according to the course makeup. For example, in CS1 at NCSU, the weekly paired lab assignments count for only 10% of the overall grade. In the software engineering course at NCSU, the students must maintain a passing grade on individual work to pass the class.

Observations: At Virginia Tech, pair programming was used as a more "individual" approach to software development in what is generally a groupwork-dominated course. Sixty percent of the course grade is group related, with the group project worth 40% of the grade, group activities worth 10%, and participation 10%.

The pair programming phase of the group project required every student to design and program a unique component of the project, a definitive skill of the course and discipline. The pairs were established outside the groups; that is, pairs of students who were also group members were disallowed. This method for pairing ensured that each group member had a working knowledge of the earlier project phase results, such that each individual could implement some portion of the group project's interface. However, the pair programming ensured that there was support for the students in their programming efforts. In the end, though, each student received an individual grade, shifting toward a more equal balance of group-individual grades.

4.6. Choosing the pairs

Guideline 6. *When assigning pairs, instructors should attempt to maximize the chances students will work well together.*

At NCSU, we assign the pairs rather than allow them to choose their partners. Through the PairEval system, we monitor the compatibility of our pairs in addition to their contribution. We have found that less than 9% of pairs report compatibility problems [14]. Students consistently express a desire to work with a partner of equal or better skill level relative to themselves. We have examined a variety of factors to determine if we can proactively assign effective student pairs. We have found that pairing a Myers-Briggs Sensor with an Intuitor has resulted in pairs that are statistically more likely to rates themselves "Very Compatible" compared with other partner matchings. Dysfunctional pairs appear to be the result of matching students of vastly different skill levels and work ethic [14].

Observations. The Virginia Tech students used the PairEval system to assess their personality types, experience, and partner preferences. Based upon the compatibility information collected with PairEval, pairs were selected to help ensure that participants would be matched in terms of personality types.

PairEval seemed to facilitate the creation of good matches—pairing excited, talkative students with calmer, more laconic ones. This arrangement was often to the benefit of information exchange but sometimes with some unspoken tension where the "calm" partner clearly wants to complete the project with a minimum of talking. However, this is only an observation; there were no complaints about this, save for one person who commented about the partner being too "quiet and professional." By the second pairing, students seemed to settle into their assignment more quickly.

4.7. Pair rotation

Guideline 7. *Students should have different partners throughout the semester.*

At NCSU, we assign students new partners at least three or four times per semester. Periodically assigning new partners is beneficial for the students because they have the opportunity to meet more of their peers. Also, students will be less likely to be intolerant of their partner if they know their "relationship" only lasts a week or two. Rotating pairs is beneficial for the teaching staff because obtaining multiple forms of peer evaluation on each student provides a more accurate picture of the contributions of the student. Additionally, the regular pair rotation allows dysfunctional pairs to separate without overt action on the part of the instructor.

Observations. At Virginia Tech, the students rotated pairs once during the four sessions, after the first two pair programming sessions, so that students would have the opportunity to meet more of their peers and to benefit from their added knowledge and expertise.

4.8. Students must promptly notify instructors of problems

Guideline 8. *Students must understand that problems with their partner must be surfaced immediately to give the instructor a chance to correct the situation.*

The most common problem with pairing is non-participation on the part of one student. Students are encouraged to report problems with their partner as quickly as possible. The instructor must then work to understand the non-participatory student's perspective as well, and/or ask the student if he or she intends to continue with the class. If the instructor determines that a student is not going to contribute fairly to an assignment, then the partner will be given reparations on the assignment, such as the option to complete a subset of the assignment individually. When students report problems at the last minute, they are not given these same reparations, motivating the students to report problems early when corrective actions are still possible.

Observations. The pairs seemed to work together well in the Virginia Tech implementation of the project, with no serious complaints emerging from any of the four sessions. As the pair programming session was well into the course syllabus—the ninth and tenth weeks of a 15-week semester—the less serious students had already withdrawn from the course. Also, the course is generally taken by juniors and seniors who have previously demonstrated a level of responsibility and an ability to work well with others.

4.9. Pair programming ergonomics

Guideline 9. *Pairs should be able to comfortably sit next to each other as they work, and both should have easy access to the monitor, mouse, and keyboard.*

At NCSU, the software engineering laboratory has an ideal pair programming setup. All computers in the lab have two monitors, two mice, and two keyboards and the room is arranged as shown in Figure 1 to facilitate pair-to-pair communication. A more traditional setup is adequate if two people can sit comfortably next to each other where both can see the display (generally a six-foot table per pair). The litmus test is that the driver and navigator should not need to switch chairs when they switch roles.

Observations. At Virginia Tech, there was no laboratory available during the class time, so we required all students to bring their laptops to class for the pair programming sessions. The situation was workable for four sessions, but for more sessions we would have pursued a lab setting that better promotes optimal information exchange.

5. Impact on Guidelines Based on Virginia Tech Experiences

Based upon the Virginia Tech experience, we suggest one modification and two additions to our guidelines. First we modify Guideline 4 to the following:

Revision to Guideline 4. *When students are pair programming outside of a closed laboratory or classroom setting, instructors should provide a systematic mechanism for obtaining students' feedback about their partners and must act upon the feedback when indications are a student is not being an equal participant.*

If the teaching staff is actively engaged in all pairing activities, the staff can perform first-hand intervention on ineffective or problem pairs, thus reducing the need for peer evaluation. The Virginia Tech teaching staff did not make use of the pair evaluation feedback and the experience was mostly positive. The CS1 class at NCSU also does not collect or follow up on peer evaluation feedback. Similar to the Virginia Tech class, the NCSU CS1 class does not require pairing outside of class, and pairing is only done by the student under the watchful eye of a TA. However, peer evaluations should always be conducted when any paired activity occurs outside of the teaching staff's observation

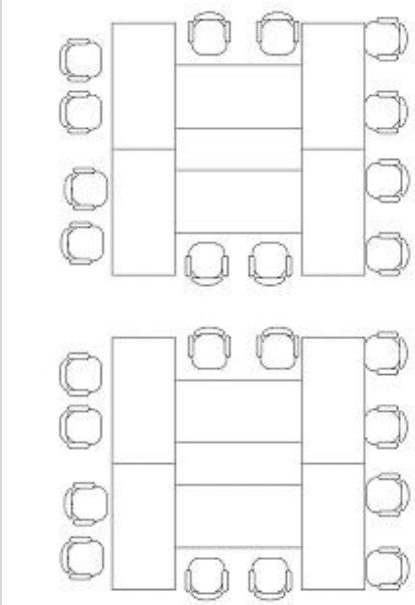

Figure 1: Pair Programming Lab at North Carolina State University

Additional Guideline 10. *The programmers in a pair should be working toward a common goal.*

In the Virginia Tech class, the two students in a pair each had individually graded assignments to complete with the help of his or her partner. Therefore, when pairing, the students were not working toward a common goal but were assisting each other in complete individual goals. The students in a pair were not on the same project team. The motivation for this arrangement was to ensure that each student has a working knowledge of the earlier project phase results of their team to be able to implement some portion of the group project's interface without another team member. However, some of the communication and motivation problems may have been reduced had the two students been working toward a common goal.

Additional Guideline 11. *Teaching staff should encourage pairs to find answers on their own rather than providing them with answers.*

The TA at Virginia Tech often would provide a general direction and would redirect questions to the pairs in hopes of helping them identify ways to find the answers themselves. In doing so, he could help the students gain confidence in their ability to work independently and to learn about searching for and finding answers. When students work alone, such question redirection might be frustrating. But, in pairs, students generally have good luck figuring out how to solve their problems together, particularly with general guidance provided by the TA.

6. Summary

Pair programming has been shown to create an environment conducive to active learning and collaboration, to help lower student frustration with challenging problems, to increase programming self-confidence, and to increase interest in information technology [1]. In this paper, we present some guidelines for classroom management when pair programming is used in the classroom. These guidelines were developed through the experiences of over one thousand NCSU students collaborating on software development in courses over the last seven years. We compare these guidelines to the experiences of a Virginia Tech HCI class in which pair programming was used. The students in the HCI class pair programmed during four 75-minute class periods, in the classroom that was monitored by the TA. The Virginia Tech class followed six of the NCSU guidelines. They did not follow up on peer evaluation; have a need to have students promptly report pairing problems; or have an ergonomic pairing setup. Because these students only worked in the classroom with their partners for a short period of time, none of these turned into problems. As a result, we evolved the guideline that prescribed that peer evaluation was an important component for success of pairing in the general sense. Their experience and

that of the NCSU CS1 class suggests that perhaps such peer evaluation is only necessary when pairing is also done outside of the classroom.

We also added two new guidelines. The first suggests that it is important for students in a pair be working for a common goal. The second additional guideline suggests that teaching staff refrain from telling pairs the answer and instead point them in the right direction and enable them to find the answer together. We hope these guidelines can help other educators be as successful as possible with pair programming in the classroom.

7. Acknowledgements

We would like to acknowledge the instructors and teaching assistants at North Carolina State University who have helped to institute pair programming: Carol Miller, Suzanne Balik, Ed Gehringer, Matthias Stallman, Mark Sherriff, Sarah Smith Heckman, Kristy Boyer and many others. This material is based upon the work supported by the National Science Foundation under Grants ITWF 00305917, BPC 0540523, and STTR 0740827. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7. References

[1] S. B. Berenson, K. M. Slaten, L. Williams, and C.-w. Ho, "Voices of Women in a Software Engineering Course: Reflections on Collaboration " *ACM Journal on Educational Resources in Computing,* vol. 4, no. 1, pp. 1-18, March 2004.

[2] S. B. Berenson, L. Williams, and K. M. Slaten, "Using Pair Programming and Agile Development Methods in a University Software Engineering Course to Develop a Model of Social Interactions," in *Crossing Cultures, Changing Lives Conference*, Oxford, UK, 2005, p. to appear.

[3] J. Bevan, L. Werner, and C. McDowell, "Guidelines for the Use of Pair Programming in a Freshman Programming Class," in *Conference on Software Engineering Education and Training*, Kentucky, 2002, pp. 100-107.

[4] B. Hanks, "Problems Encountered by Novice Pair Programmers," in *International Computing Education Research Workshop*, Atlanta, GA, 2007, pp. 159 - 164.

[5] A. Joseph and M. Payne, "Group Dynamics and Collaborative Group Performance," in *Thirty-fourth SIGCSE Technical Symposium on Computer Science Education*, Reno, NV, March 2003, pp. 368-371.

[6] D. B. Kaufman, R. M. Felder, and H. Fuller, "Peer Ratings in Cooperative Learning Teams," in *American Society for Engineering Education*, Charlotte, NC, 1999.

[7] L. Layman, "Changing Students' Perceptions: An Analysis of the Supplementary Benefits of Collaborative Software Development," in *19th Conference on Software Engineering Education and Training (CSEE&T '06)*, Turtle Bay, Hawaii, 2006, pp. 156-166.

[8] L. Layman, L. Williams, J. Osborne, S. Berenson, K. Slaten, and M. Vouk, "How and Why Collaborative Software Development Impacts the Software Engineering Course," in *Frontiers in Education*, Indianapolis, IN, 2005, pp. T4C 9-14.

[9] N. Nagappan, L. Williams, M. Ferzli, K. Yang, E. Wiebe, C. Miller, and S. Balik, "Improving the CS1 Experience with Pair Programming," in *ACM Special Interest Group Computer Science Education (SIGCSE) 2003*, Reno, 2003, pp. 359 - 362.

[10] D. Oblinger, "Boomers, Gen-Xers, and Millennials: Understanding the New Students," *Educause Review,* vol. 38, no. 4, pp. 37-47, July/August 2003.

[11] B. Simon and B. Hanks, "First Year Students' Impressions of Pair Programming in CS1," in *International Computing Education Research Workshop*, Atlanta, GA, 2007, pp. 73-86.

[12] K. M. Slaten, M. Droujkova, S. Berenson, L. Williams, and L. Layman, "Undergraduate Student Perceptions of Pair Programming and Agile Software Methodologies: Verifying a Model of Social Interaction," in *Agile 2005*, Denver, CO, 2005, pp. 323-330.

[13] L. Williams, "Lessons Learned from Seven Years of Pair Programming at North Carolina State University," *Inroads: ACM SIGCSE Bulletin,* vol. 39, no. 4, pp. 79-83, December 2007.

[14] L. Williams, L. Layman, J. Osborne, and N. Katira, "Examining the Compatibility of Student Pair Programmers," in *Agile 2006*, Minneapolis, MN, 2006, pp. 411-420.

[15] L. Williams, C. McDowell, N. Nagappan, J. Fernald, and L. Werner, "Building Pair Programming Knowledge Through a Family of Experiments," in *International Symposium on Empirical Software Engineering (ISESE) 2003*, Rome, Italy, 2003, pp. 143-152.

[16] L. Williams, E. Wiebe, K. Yang, M. Ferzli, and C. Miller, "In Support of Pair Programming in the Introductory Computer Science Course," *Computer Science Education,* vol. 12, no. 3, pp. 197-212, 2002.