JRTE, *42*(1), 29–49

Pair Programming in Middle School: What Does It Look Like?

Linda Werner
University of California, Santa Cruz

Jill Denning
Education, Training, Research Associates

Abstract

Few early intervention efforts have improved the representation of women in computer science and engineering (CSE) disciplines, but pair programming has shown promise for reducing gender differences among college students. The current study is the first to examine this promising practice in middle school. In an effort to better understand what pair programming looks like, we describe an observational study of middle school girls. We coded audiotape transcripts to show the kinds of interactions that appear to promote or undermine effective problem solving. The findings are interpreted in terms of how to promote the kinds of interactions that make it more likely that middle school students will persist in the kind of problem solving that will prepare them for further CSE coursework. (Keywords: pair programming, gender, collaborative learning, problem solving)

INTRODUCTION

The gender difference in computer access that was seen in the 1980s and early 1990s has all but disappeared, whereas the gender difference in enrollment in baccalaureate programs of computer science and computer engineering (CSE) persists (Barker & Aspray, 2006). The Committee on Equal Opportunities in Science and Engineering (2004) cites the importance of attracting, retaining, and increasing student diversity in science, technology, engineering, and mathematics (STEM) disciplines. In non-CSE STEM disciplines, gender is not a factor in the choice of major when students' grades and patterns of course taking are considered. Because of this, gender parity is the trend for all STEM disciplines except CSE (Strenta, Elliott, Adair, Matier, & Scott, 1994). Cohoon and Aspray (2006) list two factors that contribute to the ongoing underrepresentation of women in CSE: "an inadequate understanding of the underlying and immediate causes" of the underrepresentation and "inadequate intervention efforts" (p. 137).

Some have suggested that one underlying cause of women's underrepresentation in CSE is a competitive and masculine culture. Cohoon and Aspray (2006) suggest that "computing culture is masculine," but they question whether the computing culture causes overrepresentation of males or whether the presence of a disproportionate number of males in computing give it a masculine culture. Male culture characteristics, according to the Bem sex role inventory that Cohoon and Aspray (2006) reference, include competitiveness and dominance. Studies have found that perceptions of IT work as competitive

and socially isolating dissuade girls from pursuing IT work (American Association of University Women Educational Foundation Commission on Technology, 2000).

In an attempt to determine whether creating a culture of collaboration would increase retention of women in CSE, Werner and her colleagues studied pair programming among university students (McDowell, Werner, Bullock, & Fernald, 2002; McDowell, Werner, Bullock, & Fernald, 2006; Werner, Hanks, & McDowell, 2005). Pair programming, as the name implies, involves two students sharing one computer (Barker & Cohoon, 2006; Barker & Cohoon, 2007–2008; Williams & Kessler, 2000), and there are clear roles: one is the driver who works the keyboard and mouse while the other navigates. Successful use of pair programming requires collaboration rather than competitiveness and dominance. Werner et al. (2005) found that the benefits of pair programming included both performance and persistence in computer science–related majors among both male and female students. Our research and research by others suggest that working with a partner in a collaborative way is appealing to girls, because it is aligned with their social interests (Denner, Bean, & Martinez, 2009; Kekelis, Ancheta, Heber, & Countryman, 2005). Similarly, Barker, Snow, Garvin-Doxas, and Weston (2006) state:

> ... sixth-grade girls are more focused on becoming eighth- or ninth-grade girls than they are on becoming adults with careers, and given the immediate need of ensuring that college-bound girls complete high school course work that will reduce the experience gap they face as first-year college students, more research is needed on discovering approaches that will work to convince girls that taking those courses is an appropriately feminine pursuit. (p. 131)

A second underlying cause of women's underrepresentation is low confidence in solving problems on the computer. Recent reports point to problem solving as a fundamental skill in a digital age. For example, Wing (2006) describes the importance of computational thinking, which she defines as "reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation" (p. 33). Problem solving is a key part of computer science and engineering. However, Margolis and Fisher (2002) found that a major reason women drop out of college computer science courses is a lack of perceived confidence in solving problems. Some research suggests that girls are less likely than boys to persist in the face of problems or challenges in math and science when they believe that intellectual skills are a gift (Dweck, 2006). Similarly, others find that girls attribute success while working with computers to luck and attribute failure to lack of ability, whereas the opposite is true for boys (Cooper & Weaver, 2003). Girls who view success as a function of luck are less likely to work hard to solve problems. But we know of no studies that describe what computer-based problem solving among middle school students looks like, nor of the process through which interactions with others either undermine or promote girls' confidence to figure things out on the computer.

In our current research, we add to the body of knowledge about pair programming and problem solving. In another paper (Denner & Werner, 2007), we summarize the kinds of problems girls say they encounter, and their strategies for addressing them. In this paper, we focus on how middle school girls pair program, because little is known about what happens when girls are problem solving in pairs on the computer while creating information technology. To address this gap, we use audiotape data from an after-school and summer program in which middle school girls make a computer game with a peer. This report describes the only known study of pair programming a game at the middle school level. In this part of the study, we are specifically looking for answers to the following questions:

- What does pair programming look like for middle school girls?
- What kinds of pair interactions promote or undermine problem solving for middle school girls?

Research on Problem Solving and Collaborative Work at the Computer

Little is known about how students problem solve at the computer while working with a partner. Studies of children playing together at the computer, but not programming or creating information technology, suggest that effective collaboration helps them perform better, build confidence, and increase motivation to continue playing (Inkpen, Booth, Klawe, & Upitis., 1995; Littleton & Light, 1998).

Werner and colleagues studied pair programming and found that university students who worked with a partner in their introductory computer programming course were more confident in their solutions, declared computer science–related majors more frequently, passed this and subsequent programming course at higher rates, and produced higher-quality computer programs than students who worked solo (McDowell et al., 2002; McDowell et al., 2006; Werner et al., 2005). These results were true for both women and men students; however, the gender gap for increased confidence in program solutions narrowed because the 24% increase in confidence in the programming solutions for women who pair programmed was so much greater than the 15% confidence boost that pairing gave the men. These results were independent of the partner's gender. Pair programming is particularly promising for educational programs in K–12 because it encourages peer scaffolding, clear roles, and frequent feedback.

Pair programming offers opportunities for students to engage in metacognitive acts and to build the collaborative aspects of problem solving that are fundamental parts of persisting in CSE disciplines. Cognitive science research shows the importance of metacognition for the transfer of knowledge to new situations (Bransford, Brown & Cocking, 1999). Programming with a partner requires students to perform operations that involve metacognition. For instance, students must summarize and explain what they know, respond to immediate feedback, take time to work through what they do not understand, and ask questions—all high-level thinking skills that foster self-monitoring and improve performance (Palincsar & Brown, 1984). To work together effectively,

students must describe their reasoning and ask thoughtful questions, identify and explain contrasting answers, and do enough perspective taking to provide ongoing assessment, feedback, and support (Fuson, Kalchman, & Bransford, 2005; Kafai, 1995; Kafai, Ching, & Marshall, 1997; Rogoff, 1998; Schwartz & Bransford, 1998). This is true when working with a less experienced partner (Rogoff, 1998). The learning process for the partner with less computer experience entails the integration of new knowledge with prior understanding and allows them to participate in activities that they could not do alone (Rogoff, 1990). Working in a pair can enhance students' metacognition if they monitor what they learn as they give advice, act as a critic of their partners' thinking, resolve conflicts, and correct mistakes (Goos, Galbraith, & Renshaw, 2002; Roschelle, Pea, Hoadley, Gordin, & Means, 2000). This research suggests that collaborative learning can increase metacognition when students justify their reasoning, learn from others' mistakes, and receive critical feedback. However, most of this research focused on collaboration that was off the computer or involved using rather than producing software. Clements and Nastasi (1999) state that metacognition includes three component parts: performances, knowledge acquisition, and metacomponents. In this paper, we are primarily interested in metacomponents because these deal with the executive-level processes of problem solving and are similar to Goos and colleagues' metacognitive acts.

Not all pairings are productive, but there is some research on ways to increase the likelihood that pair programming will result in effective problem solving. Regardless of prior experience, effective pairs must establish a shared social reality or joint problem solving space and create a dialogue to foster new, collaborative ideas (Rogoff, 1990; Roschelle & Teasley, 1995). Partners are more likely to articulate and internalize what they are learning when they are on equal social levels and are friends (Azmitia & Montgomery, 1993; Miell & MacDonald, 2000). This is particularly important for collaboration on an open-ended, creative task (unlike tasks in which there is a single "right" answer) because friends know how to develop a shared space and generate ideas together (Miell & MacDonald, 2000). Azmitia gives three reasons for choosing partnerships based on friendships. First, for complex tasks, collaborations are longer lasting when they are between friends rather than between acquaintances. Second, problem-solving benefits are derived because friends' collaborations use dialogues that build on each other's reasoning. Third, friends are more willing to increase the role of the novice in friendship-pairs of unequal skill.

Pair programming appears particularly promising for increasing learning and engagement with technology among students with limited computer experience. Studies of problem solving and task performance find that girls and students from economically disadvantaged communities benefit the most from working together (Light, Littleton, & Bale, 2000; Perret-Clermont, Perret, & Bell, 1991). Because working with a partner may also result in social support, it is a particularly promising strategy for improving the confidence, problem solving, and performance of students who do not typically persist or excel in computer science, such as girls (Margolis & Fisher, 2002; Nelson, 1993; Werner et al., 2005).

METHOD

Participants

One hundred twenty-six girls were voluntary participants in an after-school and summer program called Girls Creating Games (GCG). They all live in a small urban community in Central California and range in age from 10 to 14 years (M=11.75, SD=1.0). The girls' self-reported ethnicity was mostly white (58%) and Hispanic/Latina (31%); 35% of all girls reported that they speak a language other than English at home at least some of the time. We collected data for this part of our work from 10 pairs of girls. We ran our program six times from spring 2003 to fall 2004. All pairs in the last cohort were audiotaped if both partners were present on the day we taped. The mean age for this subgroup is 11.45 (SD=0.61). These girls' self-reported ethnicity was mostly white (62%) and Hispanic/Latina (29%); 43% of these girls reported they speak a language other than English at home at least some of the time. All of these girls reported they have a computer at home they can use.

Program Description

A detailed description of the program can be found in another paper (Denner et al., 2005). The goal of the program is to increase middle school girls' interest, confidence, and ability to pursue courses and careers in information technology. The girls were given instruction in both Macromedia's Flash™ MX software and in pair programming to design and create computer games. The following description of our program has been published elsewhere:

Participants in this program create computer games, which are interactive story narratives, with Macromedia's Flash™ program. The idea for the game structure came from a series of books in the 1980s called "Choose-Your-Own-Adventure," where readers select a path at key decision points in the story to create their own series of events. Based on this model, the participants work in pairs to write stories about themselves or their interests and produce Flash games to tell the stories. The final products are posted on the Internet. … each individual story and its organization is unique (Youthlearn Institute at EDC, 2007).

The GCG program provided the opportunity to develop, test, and refine instructional strategies to support children to work in pairs. Instructional strategies include role modeling effective and ineffective pair programming, from which the girls identified pair programming rules. The role-modeling activity consists of two scripts, one each for *bad* and *good* pair programming behaviors, performed by the program leaders. When done with both, the students identified pair programming rules. These participant-identified rules were built into a poster, signed by the students, and displayed on the wall of the workspace (Werner et al., 2004). There was also public recognition of effective pair programmers each week. A video of the scripts has been identified as a "recommended resource" by the Computer Science Teachers Association and is available, along with the lessons, to download at http://psweb.etr.org/gcgweb (Werner & Denner, 2006).

Instruments

The data come from 8 of the 10 audiotape transcripts of pairs working on their computer games and were collected during the last of six implementations of the program. The volume and quality of two of the tapes made the dialogue unintelligible.

Excerpts from the audiotape transcripts are used to show what it looks like when pairs of girls tried to solve problems themselves, and what led up to them asking for help or using other resources. The transcripts are conversations between two girls working together, with one student designated as "A" and the other as "B." In one transcript, there is a "T" designating the teacher. There are references to instructional materials provided by the GCG program, including flowcharts of the steps needed to perform a task in Flash.

Procedure

We collected the audiotape transcripts one time with each pair of girls during the last semester of implementation, at approximately week 6 of the 12-week program. We taped the pairs of girls for approximately 30 minutes while they worked at their computers on their games. We transcribed and coded the tapes to describe their metacognition and the nature of their collaboration when faced with a challenge, as well as how interactions support or undermine persistence.

Following other studies that use discourse to understand problem solving, we identified episodes that represent different kinds of problem-solving behaviors. To describe the role of metacognition and collaboration in pairs' problem solving, we annotated the transcripts for each conversational turn using a coding scheme based on those developed by Goos and colleagues (Goos & Galbraith, 1996; Goos, Galbraith, & Renshaw, 2002) and Clements and Nastasi (1999) for their work with Logo. Goos and colleagues described metacognitive acts as either new ideas (when new information was identified or a strategy is suggested) or assessments of the strategy (assessment of the appropriateness of the strategy or understanding of the results). We coded the transcripts to reveal aspects of metacognition, using Clements and Nastasi's definition of metacomponents: problem identification, strategy choice (choice and sequencing of steps of solution), and monitoring and assessment of strategy. From these two basic approaches, we coded the transcripts for the metacognitive acts of problem identification, strategy choice with the new idea used when the strategy choice is first mentioned, and assessment of result.

To highlight the range of collaborative problem solving, we use an additional coding scheme from Goos et al. (2002) that is based on approaches by Vygotsky (Teasley, 1997). This scheme helps differentiate successful and unsuccessful problem-solving interactions. These coding categories are self disclosure, a category of statements that clarify, elaborate, or justify a person's thinking; feedback request, a category of questions that invite a partner to give feedback about one's own thinking; and other monitoring, a category of questions, statements, and responses that aim to understand their partner's thinking instead of their own thinking.

Codings of how interactions support or undermine persistence in the face of challenges are based on the work of Wegerif and Mercer (1996). According to

their definitions, disputational talk consists of disagreements and individual decision making, cumulative talk is when partners are positive but uncritical and construct a common understanding, and exploratory talk consists of challenges and suggestions where knowledge and reasoning processes are made visible. Additionally, we include transcribed extracts of students' talk so that readers can learn from our interpretations and comment (Mercer & Wegerif, 1998).

RESULTS

Interactions that Support Persistence in the Face of Challenges

Below we present sections of four transcripts that illustrate common patterns in girls' interactions as they addressed challenges at the computer. We feel these examples are representative of positive interactions that support persistence.

The first transcript below (Example 1, pp. 36–37) is an example of the problem-solving steps often used by those who are effectively learning to program computers:

- Identify a problem
- Use documentation to determine if the instructions are followed correctly
- Compare the nonworking instance to a working instance to determine if there are differences that can shed light on the source and cause of the problem
- Tinker
- Ask an expert

This type of talk fits within Wegerif and Mercer's (1996) definition of exploratory talk.

The transcript shows how the conversation moves back and forth between metacognitive acts (italics) and efforts to collaborate (bold) as the students attempt to work through challenges. In this excerpt, Partner A is the navigator and Partner B is the driver, and they are trying to figure out how to add music to their game. Initially, they do not refer to examples where they already made the music work, and they do not tinker, but instead first attempt to resolve their problem by using the flowchart guide. Partner A reads out loud the information from the flowchart about how to add music to their game. When they first get stuck (move 71), they digress (move 75). Partner B brings them back to task by monitoring her partner's thinking with short affirmations (moves 76, 78, and 80) and then proposing a new idea (move 80). This results in a series of collaborative moves and tinkering.

When Partner B notices that the music in their game is not working (move 88), this leads to a series of metacognitive moves, including using debugging skills in an attempt to problem solve. First she suggests they look at another instance of the use of music and see the difference between the two, drawing on previous experience to solve their problem. However, before they can debug, Partner B must both convince herself and her partner that there is a problem (move 95). This involves using metacognition, when they refer to an example

Example 1

57. A: Let's like…

58. B: Choosing here. So can we…I forgot how to add music (p*roblem identification;* **self-disclosure**). Can you go to the… (**other-monitoring;** *strategy choice/new idea to look at the flow charts*)?

59. A: Sure, I'll go to the flow charts (*assessment).* Hey, that's mine (**self-disclosure**). Yeah, put it right there.

60. B: Alright. We can look at it from an angle *(strategy choice).*

61. A: OK. Sounds good (**self-disclosure;** *assessment).*

62. B: So, if we're going to be adding, adding music to a scene (*strategy choice).* That's what it's going to look…

63. A: Adding buttons. Scenes. Stop Action. Go to Action Script off button. Don't need that. Add sound device. OK. Start [Reading from flow chart] (s*trategy choice).*

64. B: Flash.

65. A: Add blah, blah, blah (*strategy choice).*

66. B: That's sound to it, but we have to look for C (**self-disclosure***; strategy choice).*

67. A: Oh, nah (*strategy choice).* There we go. Add to this movie. Did we start Flash (*assessment*; **other-monitoring because it is assumed the question is "Do you think we started Flash?"**).

68. B: So. Yeah. We started Flash (*assessment of result).* We need to play our music (*strategy choice).*

69. A: And…

70. B: [unclear word] a new layer of music (*strategy choice).*

71. A: We already did that (*assessment of strategy).*

72. B: We should go get her to do this? (**other-monitoring***; strategy choice/new idea to ask for help from "her").*

73. A: Okay.

74. B: Okay.

75. A: You've got a good pen (**digression; self-disclosure**).

76. B: Okay. Thank you (**digression; self-disclosure**).

77. A: I'm going to have the ones that I like. These are mine now (**self-disclosure; digression**).

78. B: All right so (**other-monitoring; digression**).

79. A: These are the ones that I use every time (**self-disclosure**). Haven't you noticed (**other-monitoring**)?

80. B: Yes (**self-disclosure**). Now let's just keep testing it. All the music (s*trategy choice/new idea).*

81. A: Why (**other-monitoring**)?

82. B: Yeah, so we can see if nobody likes it (**self-disclosure to clarify thinking**).

Example 1 (continued)

83. A: Which one we like (**other-monitoring because the implied question is "Which one do you think we like?"**). Woo, it's like I can hear. Okay. Press play. It needs to be up a little bit higher. Is that good (**other-monitoring**). Can't hear it. Okay. Yeah, that one, I like that one (**self-disclosure**).

84. B: OK choose. OK then we have to select this and drag it up (*strategy choice).*

85. A: Are you sure (**other-monitoring**)?

86. B: That's weird it doesn't work. I mean maybe (*assessment of result*; **self-disclosure; statement that clarifies one's thinking**).

87. A: OK, choose a music clip from the library [reading flowchart] (*strategy choice).* Is that where we're at? Where we're at (**other-monitoring**). OK, so go to Common Libraries. Window, Common Libraries (*strategy choice).*

88. B: Hold on, this isn't working (*assessment of result).*

89. A: OK. Choose the music list from Common Libraries. So go to Window, Common Libraries [reading flowchart] (*strategy choice).*

90. B: This isn't working (*driver repeats assessment of result).*

91. A: And then click (*strategy choice).*

92. B: Where is the music (*driver repeats assessment of result)?*

93. A: Click and drag, sound, click and drag sound to stage (*strategy choice).*

94. B: OK. That's weird it didn't work (*driver repeats assessment of result).*

95. A: Yeah, it did. It might have (*navigator assesses result).*

96. B: See on the other one (**other-monitoring**; a*ssessment of solution process and recognition of error while looking at another earlier use of music*)? That's what it did. OK, let's find the music. See look at the music (*strategy choice*). It's different (*driver assesses result;* **self-disclosure that elaborates or justifies one's own thinking**).

97. A: Oh.

98. B: See, there's a line above it (*driver assesses result;* **self-disclosure that elaborates or justifies one's own thinking**).

99. A: Oh.

100. B: I don't know what we did for that (**self-disclosure of statement about one's own thinking**)? What did you do (**other-monitoring**)?

101. A: I don't know (**self-disclosure**). When did we do that (**other-monitoring**)?

102. B: 'Cause it works. And that way it doesn't work (*driver assesses result).*

103. A: Let's ask for [the teacher's] help (*strategy choice).* Um, we can't get the music (*assessment of solution process*; **self-disclosure**).

104. B: Yeah, we still can't get the music on (*driver agrees with strategy choice and repeat of self-disclosure).*

Example 2

1. A: OK. Double click. I mean one click. Double click, one click, sorry. Now Edit. Oh, OK, Undo. I mean don't hit Undo. Click off. Now go to Black Text. OK, now go to Edit (*strategy choice).* Oh my God, how come we can't do paste in place? How come we can't do paste in place (*navigator assesses result;* **other-monitoring)?** OK. Why don't you just delete that whole thing (*strategy choice*)?
2. B: Wait, why isn't it doing it (*driver assesses result;* **other-monitoring**)?
3. A: Because you are doing it wrong (*navigator assesses strategy).* I don't know how to do it either (**navigator self-disclosure about own skills***).* Let's look on the flow chart (*strategy choice/new idea expressed by navigator)!*

of where the music did work (move 96). They discover that the two uses of music in their game appear different (see moves 96 through 99). Verbalizing this process is a way for Partner B to teach her partner about the debugging process. The two partners try to remember how they successfully did an earlier task (moves 100 through 102), but neither can remember. In two collaborative moves (103 and 104), they agree to ask a teacher for help.

In the Examples 2 and 3 (p. 39), the partners in each of the pairs first attempt to resolve their problems by talk, and then they consult the written materials (e.g., flow charts). In both of these scenarios, there are examples of how the girls build collaboration by using self-disclosure of ignorance (see move 3 of Example 2 and moves 2 and 3 of Example 3). One of the statements from Example 3 (move 3: "I don't remember how to do that") follows an earlier contradicting statement by the same partner (Partner A): "I remember how to do that" (move 1). Perhaps hearing her partner (Partner B) also admit forgetfulness allowed Partner A to move on to the next step of problem solving: "Let's go to the spec sheets" (flowcharts).

In the next excerpt (Example 4, pp. 40–41), we see two students also working well together, engaged in "cumulative talk" in which partners offer positive but uncritical comments on each other's ideas of strategy choice. Here also there are a large number of other-monitoring types of interactions between the partners as they construct a common understanding.

The data suggest that other people play a key role in how girls think about their interactions with the computer. Reasons for persisting in the face of challenges include support from another person. In Example 1, the two students worked well together and engaged in exploratory talk in which partners offer critical but constructive comments about each other's ideas of strategy choice. There are a large number of other-monitoring types of interactions between the partners in Example 1. These are interactions representing attempts to understand each other's thinking. Other kinds of supportive comments from other transcripts not shown here include: "It's not a problem; don't be sorry," and "It doesn't have to be perfect."

Example 3

1. A: Oh. Stop Action. I remember how to do that (**self-disclosure**).
2. B: I don't remember how to do that(**self-disclosure**).
3. A: I don't remember how to do that (**self-disclosure**).
4. B: How do you do a Stop Action? I forget (**other-monitoring; self-disclosure**).
5. A: OK. Let's go to the spec sheets (*strategy choice/new idea expressed by navigator*). And Down button. Go to Stop Action on Button. Oops. Stop Action on Frame. OK. So, Adding a new Layer, called Action Script.
6. B: Oh, sorry (**self-disclosure**).
7. A: What (**other-monitoring**)? No, you need to add a new layer to the Action Script (*elaboration of strategy choice from move 5*).
8. B: Huh (**other-monitoring**)?
9. A: Add a new layer (*elaboration of strategy choice from move 5*).
10. B: Oh yeah. I just.
11. A: You can name it "as" [said as word] (*elaboration of strategy choice from move 5*).
12. B: For Action Script (**other-monitoring**)? We are so cool … play…. (**self-disclosure**).
13. A: And then select Single click, Blank Key Frame. Empty Circle on Action Script. Player Timeline. So, select this. Uh-huh. Now, uh, open Actions Panel, Window or F9. OK, on F9 took longer than…. (*elaboration of strategy choice from move 5*).
14. B: OK.

In Example 2, the partners engaged in cumulative talk, which involved other-monitoring interactions with few efforts to understand the other person's perspective. Examples 2 and 3 also each have one example of how self-disclosure of ignorance during collaboration can support persistence in problem solving. As Schoenfeld (1985) has reported, sometimes the "acknowledgement of mutual ignorance" serves as a pressure-escape value, which lightens the burden of the problem-solving task.

Interactions that May Undermine Persistence in the Face of Challenges

The transcripts also provide data on why girls may not persist in the face of challenges. In Example 5 (pp. 42–43), the partners are discussing which colors to use and whether certain kinds of colors can even be used. The partners disagree and offer many commands that are characteristic of "disputational talk." Instead of consisting of elaboration, confirmations, or constructive criticisms suggesting how to solve the problem using a choice of strategy previously suggested, disputational talk contains many short exchanges consisting of assertions about a new idea for the choice of strategy and challenges or counter-assertions. Many of the moves in this excerpt are coded as *strategy choice/new idea* and *assessments*. This excerpt has a lesser proportion of feedback requests and fewer

Example 4

1. B: Oh, you made the inside of that part the outside (*assessment of result*).
2. A: Whoops.
3. B: You have to switch them (*strategy choice/new idea*).
4. A: Yeah. Like that (**other-monitoring**)? OK.
5. B: Now go to Start Over. Yeah *(elaboration of new idea of move 3).*
6. A: Okay. And lemme turn it down *(elaboration of new idea of move 3).*
7. B: And then we need, now we need like a grey…greenish color (*strategy choice/new idea).*
8. A: A grey (*assessment of knowledge;* ***other-monitoring***)?
9. B: For the...
10. A: Oh yeah, for the diamond. Like that ***(other-monitoring***)?
11. B: Wait, go up to the one below it (*strategy choice*). Yeah. So it's OK (*assessment of result*).
12. A: Should it be just like a square (*assessment of result;* **other-monitoring**)?
13. B: Sure (**self-disclosure**).
14. A: Oh, you know there's this, one thing…(*assessment of strategy;* ***other-monitoring***).
15. B: Oh, the line thing, use the line tool (*strategy choice is an elaboration*).
16. A: Hold on, there's this thing right here that we could use that we could use. Like that for that (*strategy choice/new idea*).
17. B: You want it to be a square diamond or the circle kind of diamond (*assessment of strategy*; **other-monitoring**)?
18. A: Circle (**self-disclosure**). Let's do that (*strategy choice*).
19. B: Oh. Okay (**self-disclosure**). It's kind of small; you have to make it bigger (*assessment of result).*
20. A: Wait how do you do that (**other-monitoring**)? Oh, right, got it (**self-disclosure**).
21. B: It's purty-ful (**self-disclosure**). I'm trying to make a diamond ring (**self-disclosure**).
22. T: Oh, yeah, it's going to be kind of like (*strategy choice)…*
23. B: Yeah, and we have to make it (*strategy choice*). Oh, I know! I know (**self-disclosure**)!
24. T: One other thing! Sorry I keep making you guys do this**.** But, um, for each of those extra scenes that I made you make (*strategy choice/new idea*)?
25. A&B: Yeah.
26. T: For all of them, do you put your Stop Action script on each of them (**other-monitoring**)?
27. B: Not yet (*assessment of strategy*).
28. A: Oh, we didn't (*repeat of assessment;* ***self-disclosure***).

Example 4 (continued)

29. B: Oh, wait, yeah, we did *(assessment of strategy;* **self-disclosure**).
30. A: Yeah, yeah (*repeat of assessment*).
31. T: Stop Action script on every single one (**other-monitoring**)?
32. A: No, not every (**self-disclosure**). Can you do that right now (**other-monitoring**)?
33.B: Yeah.

instances of other-monitoring than seen in the earlier excerpts. Although there are many self-disclosures, they are not linked to finding out what the other person thinks or elaborating on their ideas. There appears to be little self-disclosure of ignorance other than in move 3.

The previous section had examples of two students working well together. In Example 1 they were engaged in exploratory talk in which partners offer critical but constructive comments on each other's ideas of strategy choice; in Example 4 they were engaged in cumulative talk in which partners offer positive but uncritical comments on each other's ideas of strategy choice. There are a higher proportion of other-monitoring types of interactions between the partners in Example 1 and between the partners in Example 4 than between the partners in Example 5; other-monitoring types of interactions represent attempts to understand the partner's thinking and create a common understanding. The partners in Example 5 are not making many attempts to understand the partner's thinking in either a critical or uncritical way until Partner A switches to the role of driver (move 33). She asks to be the driver because she thinks she "know(s) how to do this." From this point, the talk seems more characteristic of exploratory talk; in moves 34 and 36 Partner B assesses the strategy of the use of multicolored words (moves 34 and 36), and Partner A assesses this same strategy (move 37).

How Pair Programming Motivates Girls to Persist

Efforts to understand and mediate the underrepresentation of women in CSE disciplines have been hampered by a lack of research on their interactions while programming. In an attempt to fill this gap, the current paper analyzes how young women interact during pair programming while building a computer game. We show that a coding method can be used to identify interaction processes while pair programming. The findings provide some indication of how middle school girls work on the computer with a peer, their communication patterns, as well as the strategies they use to address challenges while working with a peer. They provide some insight into the types of interactions that promote or undermine the kind of problem solving that will help girls persist in solving their problems while creating IT. Both experience and confidence with this type of problem solving is likely to increase the chances they will pursue and persist in CSE disciplines.

Example 5

1. B: Yah! I don't like that color (**self-disclosure**)!
2: A: I like that color (**self-disclosure**).
3. B: Well, I can't make them like that (**self-disclosure**).
4. A: Make them like that (*strategy choice/new idea*).
5. B: OK. How about, I wanna, um…oh (*strategy choice/new idea*).
6. A: You want Technicolor on it so, n-n-no, stop, stop, stop (**other-monitoring**).
7. B: I tell you what, we'll just try it right here, then we can move it off it (*strategy choice/new idea*).
8. A: No. Let's not (*strategy choice/new idea).* Oh, my gosh. Not again. Oh, oh (*assessment of result*).
9. B: Why is it doing that (**other-monitoring**)? It's not letting me (*assessment of result*).
10. B: It doesn't have to be on any layers.
11. A: So add layers. Here. So go in once. Yeah. And now (*strategy choice/new idea*).
12. B: Oh! This is not good (*assessment of result*).
13. A: So go just pick another color (*strategy choice/new idea*).
14. B: Well it's not responding (*assessment of result*). I like this color that just kind of jumps out at ya (**self-disclosure and feedback request because of question**)?
15. A: OK, fine (**self-disclosure**).
16. B: Blue or red (**other-monitoring**)?
17. A: Blue (**self-disclosure**).
18. B: Blue, blue. Oh, my God (*probably an assessment of result*)…
19. A: OK, here, just try another color now (*strategy choice/new idea*).
20. B: There's something seriously wrong (*assessment of result*)…
21. A: Try another color (*strategy choice elaboration of new idea of move 19*).
22. B: I'll have to try everything. We'll try it like (*strategy choice elaboration of new idea of move 19*).
23. A: Here, try…oooh, that purple is really pretty (**self-disclosure**)! Let's try it! Click. Click on there (*strategy choice*).
24. B: Whoa. Huh (*possibly an assessment of result*). No, we do grey, look.
25. A: No, do it.
26. B: Oh.
27. A: OK, just do that purple that was pretty (*elaboration of strategy choice from move 23;* **self-disclosure**).
28. B: No I really like this one (**self-disclosure**).
29. A: No keep writing like that (*elaboration of strategy choice from moves 23 and 27*).
30. B: Have chosen…needs…But one second. Dang it, that's white (*assessment of result*).

Example 5 (continued)

31. A: Oh, well.
32. B: There. Whoa. And we're just leaving it, I know how…there. This is screwed up (*assessment of result*).
33. A: OK, here, can I see the mouse for a sec **'**cause I know how to do this (**self-disclosure**). Change your color; now go right like that. See now, now it's good. And I delete that, and then *(strategy choice/new idea to change the driver*)…
34. B: I don't think you can use multicolor on words, though (*assessment of appropriateness of strategy;* **self-disclosure**).
35. A: Change the color (*strategy choice*).
36. B: Dang, why can't we use it (*assessment of appropriateness of strategy;* **other-monitoring**)?
37. A: You can't use it! Because you can't use it on that, on words. It's impossible (*assessment of appropriateness of strategy*).
38. B: Oh, yeah. Hmmm (*repeat of assessment*) ...

The data provide a description of how working in a pair can facilitate or undermine the debugging and problem solving process, which can have potentially long-term effects on females' persistence in CSE disciplines. Debugging is one of the capabilities of the Committee on Information Technology Literacy of the National Research Council's (1999) list—test a solution, find problems in a faulty use of IT, and expect the unexpected. These steps include identifying that a problem exists, isolating the source and cause of the problem, determining and performing the work necessary to solve the problem, and then testing the solution. When working with a partner, this process meant that the girls were engaged in exploratory talk, which involves metacognitive monitoring of themselves and their partners, and some of these debugging steps were evident in their interactions. Thus, by helping each other learn the debugging process, many of these pairs of middle school girls employed the problem-solving steps that will prepare them for computer science courses.

Our findings also build on the work of Schoenfeld (1985), who states:

> It seems reasonable that involvement in cooperative problem solving—where one is forced to examine one's ideas when challenged by others and in turn to keep an eye out for possible mistakes that are made by one's collaborators – is an environment in which one could begin to develop the skills that internalized, are the essence of good control. (p. 143)

Schoenfeld uses *control* to mean "selecting and pursuing the right approaches, recovering from inappropriate choices, and in general monitoring and overseeing the entire problem-solving process." We give a specific role to the navigator of the pair. That role is to oversee the "entire problem-solving process." The

findings from the current study suggest that pair programming might allow for the development of effective control strategies for solving information technology problems. Confidence gained from these experiences can be used as a foundation for other IT challenges.

As the recent movement toward emphasizing computer fluency, rather than literacy, suggests, it may be more important for students to identify and respond to challenges than to learn to use specific software programs (Committee on Information Technology Literacy, National Research Council, 1999). Pair programming may facilitate problem solving. The excerpts show the process through which different types of interactions might support or undermine problem solving at the computer while working with a peer. As Barron (2000) suggests, student interactions create representations of their problem as they attempt to solve it. To solve a problem, the partners must develop a shared understanding of what the problem is (Roschelle & Teasley, 1995). The process of negotiating this shared understanding is shown in the excerpts we presented.

The excerpts also provide information about the occurrence of joint problem solving on the computer with open-ended problems. They show examples of exploratory, cumulative, and disputational talk based on categories by Wegerif and Mercer (1996) while engaged in a design and creation task on the computer. Most previous studies examined these types of talk while using rather than creating IT, and while engaged in closed-ended problems. The excerpts also show the characteristics of girls' metacognitive activity as they design computer games in pairs, as well as the collaborative quality of girls' problem solving at the computer. Although the categories created by Goos et al. (2002) had been developed by looking at collaboration while problem solving off the computer, they were also useful for understanding computer-based collaboration.

The findings have implications for how teachers and other educators can create a learning environment in which girls develop a way of working collaboratively that will make it easier and more likely for them to persist in courses and careers creating technology. As previous studies suggest, in order for girls to persist in computer science courses and careers, they need to be resilient in the face of challenges. In successful collaborations, conversational turns build on each other and the content moves the pair closer to solving the problem (Roschelle & Teasley, 1995; Schegloff, 1991). One example is that a partner might start an idea and the other one finishes it, which is similar to what others call cumulative talk. If conflicts occur, as they often do, the pair can attempt to work with the conflict by either finding a compromise, using persuasion to convince a partner of the preferred approach, or working around the disagreement by doing something different, as shown in examples that we labeled exploratory talk. These are examples of the different techniques that can be used to negotiate and resolve challenges while working as a pair on the computer.

It is important to inform students of the range of legitimate techniques for addressing the inevitable challenges they will face while working both alone and with others on the computer. The findings suggest that girls are using a range of techniques when faced with challenges while working with a partner on

the computer to create IT. Because the audiotapes were collected from a small number of pairs at only one point in time, it was not possible to get an accurate measure of the frequency of exploratory versus cumulative versus disputational talk. Additionally, we can't determine if the girls improved their problem-solving capabilities or persistence in CSE disciplines over the length of our intervention. Although the current data cannot reveal which techniques are most effective long-term, they are a first step toward identifying practices. The next steps would be to determine which techniques and interaction styles are most effective for persistence in problem solving while creating computer games. We plan to audiotape additional pairs creating computer games at multiple points in the creation process. In addition, pre- and postcourse surveys with questions about perceptions could be useful to measure changes in interest in CSE disciplines. To succeed in technology-rich environments, students must be motivated to engage with complex materials and work with a peer. Teachers and computer-based instructional materials can support successful collaboration by encouraging both cumulative and exploratory talk that involves self-disclosure, metacognitive acts that include assessment of how strategies are working, and tinkering with new ideas.

CONCLUSION

This study addressed gaps in the research on what pair programming looks like in middle school and how it can promote or undermine effective problem solving, resulting in critical information for both researchers and educators. These findings on pair programming can inform researchers about the processes of peer interaction at the computer. Additionally, we hope that future work in this area will show that teaching specific interaction styles will result in effective problem solving while creating technology and, more important, will result in persistence in computer science and engineering disciplines.

Contributors

Linda Werner is an associate researcher and lecturer in computer science at the University of California at Santa Cruz. Her research areas include software engineering, pair programming, and social issues. (Address: CS Dept., 1156 High Street MS: SOE3, University of California, Santa Cruz 95064; E-mail: linda@cs.ucsc.edu)

Jill Denner is associate director of research at Education, Training, Research Associates. Her research areas include gender, positive youth development, and technology in education. (Address: 4 Carbonero Way, Scotts Valley, CA 95066; E-mail: jilld@etr.org)

References

American Association of University Women Educational Foundation Commission on Technology, Gender, and Teacher Education. (2000). *Tech savvy: Educating girls in the new computer age*. Washington, DC: American Association of University Women (AAUW) Educational Foundation.

Azmitia, M., & Montgomery, R. (1993). Friendship, transactive dialogues, and the development of scientific reasoning. *Social Development, 2*(3), 202–221.
Barker, L. J., & Aspray, W. (2006). The state of research on girls and IT. In J. M. Cohoon & W. Aspray (Eds.), *Women and information technology: Research on underrepresentation* (pp. 3–54). Cambridge, MA: MIT Press.
Barker, L. & Cohoon, J.M. (2007–2008). *The National Center For Women & Information Technology Promising Practices: Pair programming (case study 1): Retaining women through collaborative learning*. Retrieved on February 1, 2009, from http://www.ncwit.org/images/practicefiles/PairProgramming_RetainingWomenCollaborativeLearning_Practice.pdf
Barker, L. J., Snow, E., Garvin-Doxas, K., & Weston, T. (2006). Recruiting middle school girls into IT. In J. M. Cohoon & W. Aspray (Eds.), *Women and information technology: Research on underrepresentation* (pp. 115–136). Cambridge, MA: MIT Press.
Barron, B. (2000). Problem solving in video-based microworlds: Collaborative and individual outcomes of high-achieving sixth-grade students. *Journal of Educational Psychology, 92*(2), 391–398.
Bransford, J. D., Brown, A., & Cocking, R. (1999). *How people learn: Brain, mind, experience, and school*. Washington, D.C.: National Academies Press.
Clements, D. H., & Nastasi, B. K. (1999). Metacognition, learning, and educational computer environments. *Information Technology in Childhood Education, 10*, 5–38.
Cohoon, J. M., & Aspray, W. (2006). A critical review of the research on women's participation in postsecondary computing education. In J. M. Cohoon & W. Aspray (Eds.), *Women and information technology: Research on underrepresentation* (pp. 137–180). Cambridge, MA: MIT Press.
Committee on Equal Opportunities in Science and Engineering. (2004). *Broadening participation in America's science and engineering 1994–2003 decennial & 2004 biennial reports to Congress*. Retrieved on February 1, 2009, from http://www.nsf.gov/od/oia/activities/ceose/reports/ceose2004report.pdf
Committee on Information Technology Literacy, National Research Council (1999). *Being fluent with information technology*. Washington, D.C.: National Academy Press.
Cooper, J., & Weaver, K. D. (2003). *Gender and computers: Understanding the digital divide*. Mahwah, NJ: Lawrence Erlbaum Associates Publishers.
Denner, J. Bean, S., & Martinez, J. (2009). The Girl Game Company: Engaging Latina girls in information technology. *Afterschool Matters*, *8* (Spring 2009), 26–35.
Denner, J. & Werner, L. (2007). Computer programming in middle school: How pairs respond to challenges. *Journal of Educational Computing Research, 37*(2), 131–150.
Denner, J., Werner, L., Bean, S., & Campe, S. (2005). The Girls Creating Games program: Strategies for engaging middle-school girls in information technology. *Frontiers: A Journal of Women Studies, 26*(1), 90–98.
Dweck, C. (2006). Is math a gift? Beliefs that put females at risk. In S. J. Ceci & W. M. Williams (Eds.), *Why aren't more women in science? Top researchers*

debate the evidence, (pp. 47–56). Washington, D.C.: American Psychological Association.
Fuson, K. C., Kalchman, M., & Bransford, J. D. (2005). Mathematical understanding: An introduction. In M. S. Donovan & J. D. Bransford (Eds.), *How students learn: History, mathematics, and science in the classroom* (pp. 217–256). Washington, D.C.: The National Academies Press.
Goos, M., & Galbraith, P. (1996). Do it this way! Metacognitive strategies in collaborative mathematical problem solving. *Educational Studies in Mathematics, 30*(3), 229–260.
Goos, M., Galbraith, P., & Renshaw, P. (2002). Socially mediated metacognition: Creating collaborative zones of proximal development in small group problem solving. *Educational Studies in Mathematics, 49*(2), 193–223.
Inkpen, K., Booth, K. S., Klawe, M., & Upitis, R. (1995). Playing together beats playing apart, especially for girls. In J. Schnase & E. Cunnius (Eds), *Proceedings of computer support for collaborative learning* (pp. 177–181). Hillsdale, NJ: Lawrence Erlbaum.
Kafai, Y. B. (1995). *Minds in play: Computer game design as a context for children's learning.* Hillsdale, NJ: Erlbaum.
Kafai, Y. B., Ching, C. C., & Marshall, S. (1997). Children as designers of educational multimedia software. *Computers and Education*, *29*(2/3), 176–126.
Kekelis, L. S., Ancheta, R. W., Heber, E., & Countryman, J. (2005). Bridging differences: How social relationships and racial diversity matter in a girls' technology program. *Journal of Women and Minorities in Science and Engineering, 11*, 231–246.
Light, P., Littleton, K., & Bale, S. (2000). Gender and social comparison effects in computer-based problem solving. *Learning and Instruction, 10*, 483–496.
Littleton, K., & Light, P., & (1998). Getting IT together. In *Learning with computers: Analyzing productive interaction* (pp. 1–9). New York: Routledge.
Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. Cambridge, MA: The MIT Press.
McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education* (Cincinnati, Kentucky, February 27– March 3, 2002). SIGCSE '02. ACM, New York, 38–42.
McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. *Communications of the ACM, 49*(8), 90–95.
Mercer, N., & Wegerif, R. (1998). Is "exploratory talk" productive talk? In K. Littleton & P. Light (Eds.), *Learning with computers: Analyzing productive interaction* (pp. 79–101). New York: Routledge.
Miell, D., & MacDonald, R. (2000). Children's creative collaborations: The importance of friendship when working together on a musical composition. *Social Development, 9*(3), 348–369.
National Research Council. (1999). *Being fluent with information technology*. Washington, D.C.: National Academy Press.

Nelson, C. E. (1993). Valuing diversity in the educational process. In *Proceedings of the National Science Foundation workshop on the role of faculty from the scientific disciplines in the undergraduate education of future science and mathematics teachers* (pp. 71–74). Washington, D.C.: National Science Foundation.
Palincsar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension fostering and monitoring activities. *Cognition and Instruction, 1*, 117–175.
Perret-Clermont, A.-N., Perret, J.-F., & Bell, N. (1991). The social construction of meaning and cognitive activity in elementary school children. In L. B. Resnick, J. M. Levine, & S. D. Teasley (Eds.), *Perspectives on socially shared cognition* (pp. 41–62). Washington, D.C.: American Psychological Association.
Rogoff, B. (1990). *Apprenticeship in thinking: Cognitive development in social context*. New York: Oxford University Press.
Rogoff, B. (1998). Cognition as a collaborative process. In D. Kuhn, R. S. Siegler, & W. Damon (Eds.), *Handbook of child psychology: Vol. 2. Cognition, perception, and language* (pp. 679-744). New York: Wiley.
Roschelle, J. M., Pea, R. D., Hoadley, C. M., Gordin, D. N., & Means, B. M. (2000). Changing how and what children learn in school with computer-based technologies. *The Future of Children, 10*, 76–101.
Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. In C. O'Malley (Ed.), *Computer supported collaborative learning* (pp. 69–97). New York: Springer-Verlag.
Schegloff, E. A. (1991). Conversation analysis and socially shared cognition. In L. B. Resnick, J. M. Levine & S. D. Teasley (Eds.), *Perspectives on socially shared cognition* (pp. 150–171). Washington, D. C.: American Psychological Association.
Schoenfeld, A. H. (1985). *Mathematical problem solving*. Orlando, FL: Academic Press.
Schwartz, D. L., & Bransford, J. D. (1998). A time for telling. *Cognition and Instruction, 16*(4), 475–522.
Strenta, A. C., Elliott, R., Adair, R., Matier, M., & Scott, J. (1994). Choosing and leaving science in highly selective institutions. *Research in Higher Education, 35*(5), 513–547.
Teasley, S. D. (1997). Talking about reasoning: How important is the peer in peer collaboration? In L. B. Resnick, R. Saljo, C. Pontecorvo, & B. Burge (Eds.), *Discourse, tools, and reasoning: Essays on situated cognition* (pp. 361–384). Berlin: Springer-Verlag.
Wegerif, R., & Mercer, N. (1996). Computers and reasoning through talk in the classroom. *Language and Education, 10*(1), 47–64.
Werner, L., & Denner, J. (2006). Pair Programming Video. *CSTA Voice, 2*(2). Retrieved February 1, 2009, from http://csta.acm.org/Resources/sub/HighlightedResources.html.
Werner, L., Denner, J., & Bean, S. (2004). Pair programming strategies for middle school girls. In *Proceedings of the 7th IASTED International Conference on Computers and Advanced Technology in Education* (pp. 161–166). Kauai, Hawaii: ACTA Press.
Werner, L. L., Hanks, B., & McDowell, C. (2005). Pair-programming helps female computer science students. *Journal of Educational Resources*

in Computing 4(1), 4. Retrieved February 1, 2009, from http://doi.acm.org/10.1145/1060071.1060075
Williams, L. A., & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM, 43*(5), 108–114.
Wing, J.M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.
YouthLearn Institute. (n2007). *Model technology integration in afterschool: Girls Creating Games*, Retrieved on February 4, 2009, from http://www.youthlearn.org/afterschool/materials/GCG.pdf