Introducing Pair Programming in Intermediate C to Non-Specialists

Chris Thompson and Edwin M. Knorr *(h5c7@ugrad.cs.ubc.ca / knorr@cs.ubc.ca)*

Department of Computer Science, University of British Columbia

BACKGROUND AND MOTIVATION

Electrical and Computer Engineering (ECE) asked Computer Science to develop and deliver a mandatory second-year intermediate C programming course for electrical engineering students who are *not* in the computer engineering specialization:

- 2 credits (26 lecture hours and 12 lab hours)
- Intermediate procedural C programming techniques
- Fundamental algorithms and elementary data structures
- Introduction to scripting by invoking and using a locally installed MATLAB engine from C

We developed ***CPSC 259: Data Structures and Algorithms for Electrical Engineers*** during the summer and fall of 2011.

KEY GOALS FOR CPSC 259 LABS

1. Reinforce course learning goals
2. Inculcate effective and industry-acknowledged software development habits and methodologies
3. Foster peer instruction and collaboration while mitigating plagiarism
4. Engender student programming competence and confidence
5. Promote quick team-building

Agile Software Development

Active Learning

Collaboration and Engagement

Problem-Based Learning

A SOLUTION: PAIR PROGRAMMING

PAIR PROGRAMMING is a popular software technique in which two programmers are seated together at one workstation.

One programmer (the **DRIVER**) types, and the other programmer (the **NAVIGATOR**) performs quality control, helps point out errors (e.g., misspelled variable names, undeclared variables, out of bounds conditions), gets a better understanding of how the code fits together, etc.

The programmers switch roles on a regular basis.

PAIR PROGRAMMING is a key component of Agile Software Development, which advocates iterative development, rapid and flexible responses to change, and **lots of collaboration**.

Improved Confidence

Improved Productivity

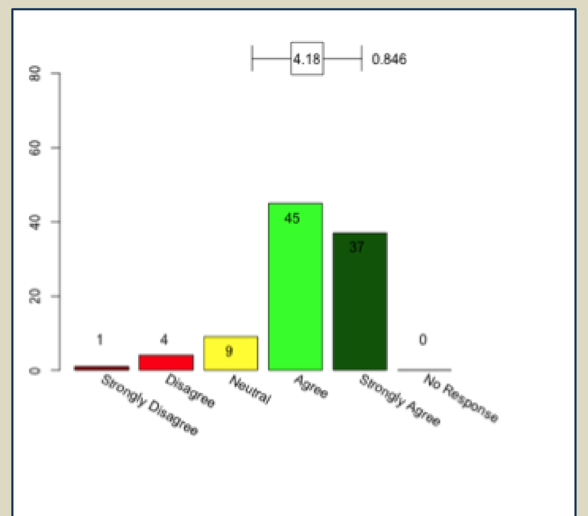
Faster Feedback

PAIR PROGRAMMING IN CPSC 259

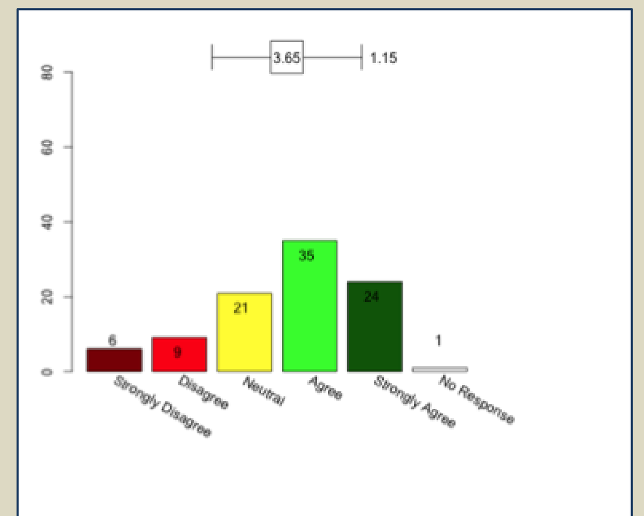
- We generated 5 two-part labs for the course:
  1. An in-lab section in which students work together but earn some marks separately
  2. A take-home section in which students work together and earn a common group mark
- We implemented a PAIR PROGRAMMING requirement in the CPSC 259 labs:
  1. Students choose their own partners, and may not work alone
  2. Students choose a new partner for each lab
  3. Students work with the same partner for both the in-lab and the take-home components of each lab
  4. Students are graded for their pair programming performance
     - Remembering to trade roles
     - Working together effectively, inside and outside the lab

A METRIC FOR SUCCESS: FEEDBACK FROM STUDENTS

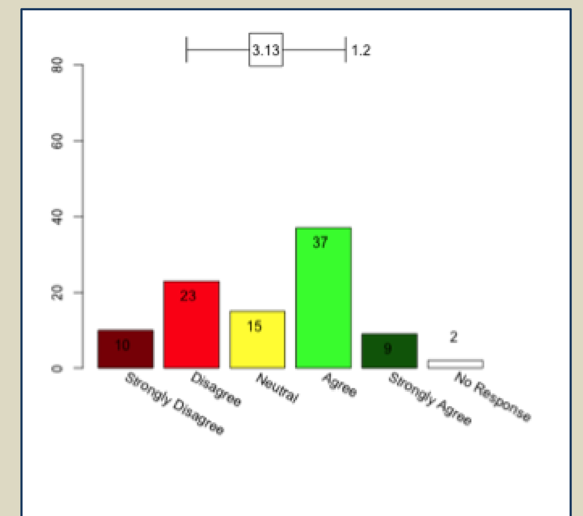
We surveyed the students at the end of the inaugural offering (January-April 2012) of CPSC 259:



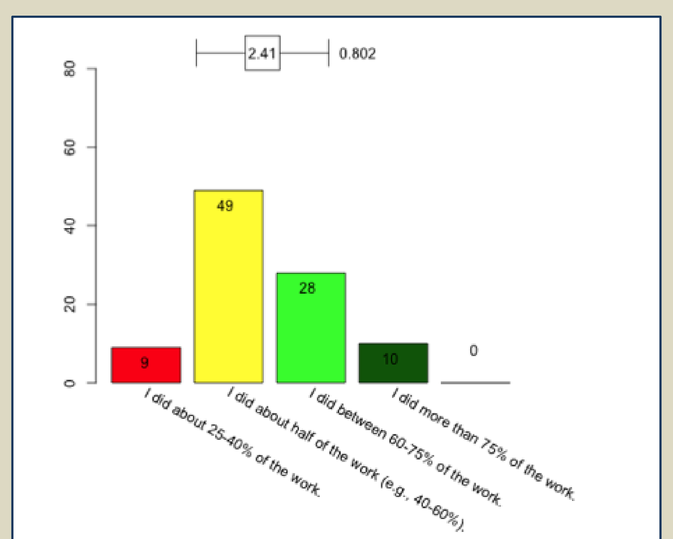
It was useful to work with a partner during the in-lab components.



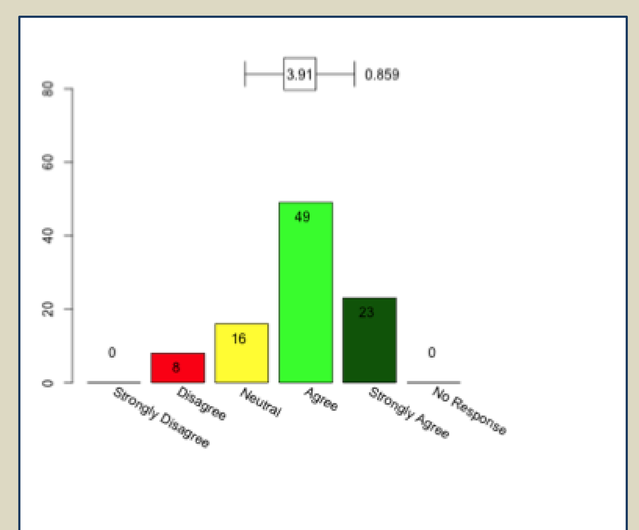
It was useful to work with a partner for the take-home labs.



I like working with a different partner for each lab.



Question 4: What percentage of a typical take-home lab was done by you, as opposed to being done by your partner?



The labs helped me to learn the material presented in the lectures.

Current work: During the second offering, we are collecting more data about pair programming and the challenges facing students in such a lab environment.

CHALLENGES AND LOOKING AHEAD

1. How can we monitor the teams in the lab in order to determine that they are PAIR PROGRAMMING correctly and effectively?

   We are employing three Teaching Assistants in each lab to provide faster and more accurate feedback (the usual CS complement is 2 TAs per lab). Are there effective alternatives?

2. How do we teach students to actively manage and nurture their partnerships?

   We generated explicit and detailed PAIR PROGRAMMING guidelines for students and provide reminders and suggestions about how to work together effectively during each lab and during office hours.

3. How should pairs be assigned?

   We decided to permit students to choose a new partner from their own lab section for each lab. Are there alternatives that will improve student success? For example, should we pair strong students with strong students? Weak students with weak students? Strong with weak? Permit students to work with anyone they wish, as often as they wish?

4. How do we manage partnerships that "just don't work"?

   Sometimes a partnership just doesn't work due to issues of: personality, availability, communication, etc. We actively manage these (rare) challenged teams on a case-by-case basis, but because we usually don't learn about the issues until well into the lab cycle, our best advice has been to continue and complete the lab solo. Are there more effective and constructive team management strategies that are appropriate for CPSC 259?

5. How do we help students clarify and respect the differences between cooperation, collaboration, and plagiarism?

SUMMARY

CPSC 259 is nearing the end of its second offering. We are employing PAIR PROGRAMMING in the lab component of the course in order to:

- Introduce students to industry-acknowledged agile software development
- Foster peer instruction, collaborative learning, and faster feedback from instructors

Developing CPSC 259 has been an iterative process. How can we respond to student performance and improve the next iteration?

This work is supported in part by: CWSEI Carl Wieman Science Education Initiative at the University of British Columbia