New Roles for Students, Instructors, and Computers in a Lab-based Introductory Programming Course

Michael Clancy
Nate Titterton
Clint Ryan
Computer Science Division
EECS Department
University of California
Berkeley, CA 94720–1776
{clancy,ryanc}@cs.berkeley.edu
nate@socrates.berkeley.edu

Jim Slotta
Marcia Linn
School of Education
University of California
Berkeley, CA 94720–1670
{slotta,mclinn}@socrates.berkeley.edu

Abstract

This paper describes our efforts to develop a new lab-based course format for computer science instruction. Building on learning science research, we created a flexible new technology platform to support students and their instructor as they participated in this new form of instruction. Students work collaboratively on Web-based activities while the instructor interacts with students in a tutorial role. The paper describes our system in detail, outlines the organization of the course that used it, and reviews and evaluates the pilot results. We then discuss the implications for computer science instruction and research in higher education.

Categories & Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education — *computer science education.*

General Terms

Design, Experimentation, Management

Keywords

CS 1, closed laboratories, online courses, distance learning, CS education research, WISE, collaboration, tutoring.

SIGCSE'03, February 19-23, 2003, Reno, Nevada, USA.

1 Background

Building on recent advances in cognitive research [1], and technology-enhanced learning environments [7], we have developed a new approach to undergraduate computer science instruction at the University of California, Berkeley. This paper presents our design of an experimental lab-based introductory computer science course and discusses our results and their implications for computer science instruction and research in higher education. Sponsored by the CITRIS project[1], our goal was to develop a technology platform that would enable students and instructors to utilize computer technology effectively in all stages of an undergraduate course. Instead of listening to lectures, students would be participating in computer-based activities (e.g., programming exercises) and instructors would be freed up to interact more closely with students as they worked in pairs or small groups.

Our system includes three components: (1) a course builder, which enables instructors to develop online activities for students, (2) a Web-based learning environment, which delivers all student activities and collects all student work in a relational database, and (3) a course portal, which serves as the learning management system for the course. (The system shares some features with a system described in [10].) We developed the prototypes of the course builder and course portal, and constructed our student learning environment by expanding on the successful platform of the Web-based Inquiry Science Environment (WISE) system [7, 9]. This technology platform ensured that each day students would show up at the lab and go directly to a set of activities that had been developed by their instructor. Activities included online discussions, programming exercises, reading of Web-delivered text, reflection notes, journal entries, and "Gated Collaborations" where students critiqued their peers' responses to a seed topic. WISE permits instructors to view student work (e.g., quiz responses and collaboration activities) in real time.

[1] The Center for Information Technology Research for Improving Society (CITRIS) is funded by the State of California and industry partners. The educational component focuses on leveraging the strengths of technology for instruction in higher education, with a particular emphasis on computer science topics.

We began by converting CS 3 ("Introduction to Symbolic Programming"), our Scheme-based introductory course for nonmajors, into a structured laboratory format that would provide the first test of this system and help us research the best designs for such a novel instructional format. CS 3 covers functional (side-effect free) programming, recursion, and use of higher-order functions for mapping, accumulating, and filtering a list [3]. Roughly ¼ of the course introduces evaluation, function definition, conditional expressions, words, and sentences. The next ¼ is spent on linear recursion. Next comes time with tree recursion and higher-order functions, along with lists in all their glory. The course ends with a relatively complex project of around 200 lines of code. The textbook is currently *Simply Scheme* [5]. Traditionally, this fifteen-week course include two hours of faculty-led lecture, two hours of t.a.-supervised closed lab, and one hour of discussion section (also led by a t.a.) each week. We sought a structured laboratory format to address the following shortcomings of the traditionally run course:

- lectures that led to "learning" by watching and listening rather than *doing*;
- attrition, especially among underrepresented minorities and women;
- student-reported "disconnects" between lecture topics, homework activities, and lab exercises;
- laboratories that did not adequately support teamwork;
- text materials that did not sufficiently engage students.

Cognitive research suggested several approaches to the design of instruction that would leverage the strengths of the technology as well as laboratory format. These include the use of case studies [2, 6] modeling the evaluation of programming code [8], and designing exercises that target challenges commonly faced by students [4].

2 Reorganization of CS 3

While enrollment in CS 3 each semester varies between 200 and 300, our first run of this course was during the summer of 2002, with 60 students initially enrolled in three sections. We are currently implementing a second round of trials with students this semester (Fall, 2002). Restructuring of CS 3 began with the decision to run an experimental section in summer 2002 with *no* scheduled lecture or discussion section—only seven hours per week of lab meetings. In the eight-week summer session, this figure doubled, so students were scheduled for fourteen hours per week of lab. This decision maximized the amount of supervised student activity, and would allow us to explore the richest possible implementation of our new technology platform in a lab-based format.

In designing the activities for our lab-based format, we began by collecting all the lab exercises, homework activities, and programs demonstrated in lecture, plus old exam problems and various other exercises. We then arranged these into chunks (each corresponding to one lab session) of content that were estimated to require three hours of student effort. Most of these lab sessions commenced with a quiz relating to the previous day's material, and continued with a step-by-step progression through programming tasks (both synthesis and analysis), perusal of adaptations from the text, and collaborative activities. Our goal was to capture all relevant content from the traditional course, and implement a new pedagogical approach that encouraged student reflection, dynamic activities (e.g., programming), student collaboration, timely reflection, and new opportunities for the instructor to interact with students. Table 1 show a day-by-day picture of the course, and Table 2 shows the organization of a portion of the second day of course activities. The code in the left-hand column of the table specifies the kind of activity, as delivered by the WISE learning environment. In "gated collaborations," students must reply to a statement, then survey replies by their peers. "Web-texts" are expository HTML pages that deliver important content, examples, etc., as designed by the instructor.

As part of this project, we developed a full-featured browser-based Scheme interpreter and text editor, based partly on the open-source, Java-based Scheme SISC.

3 Results

Enrollment in the summer course started at 60. There were three lab sections, one in the morning run by Clancy (23 students), one in the afternoon run by Ryan (22 students), and one in the evening also run by Ryan (15 students). Each section had two or three lab assistants earning credit for helping in the lab.

Quizzes and collaborative questions proved to be invaluable aids in detecting student confusion and addressing it immediately. As noted previously, the instructor could observe student answers to these questions. When most of the answers were inaccurate or incorrect, we interrupted students with a brief lecture; when only a few individuals seemed confused, we engaged these students in one-on-one tutoring sessions.

Pacing was flexible. There were occasional "catch-up" days in which not all students needed to attend class. Both the flexible pacing and the targeted tutoring served to force students to keep up; through most of the semester, the *worst* among students who attended class regularly were comparable to the *average* student in the regular academic year class.

Collaboration and discussion activities had not previously been a part of CS 3. Students generally seemed to think these kinds of activities helpful for learning. They noted the following kinds of questions as particularly beneficial:

Collaborative response

- Supply a solution to an exercise with several "right" answers.
- Supply a suggestion for understanding a given topic or technique.
- Explain *how* you got an answer.

Discussion

- Comment on mistakes you made.
- Suggest tips for understanding a given topic or technique.
- Explain productive techniques for understanding larger programs.

For some of these activities, careful reading of the text or case studies was necessary. We noticed a marked decrease in programming errors that we had attributed in the past to superficial reading of the case studies.

Students took a final exam that overlapped significantly with and was comparable in length to an exam from 1994. The fall 1994 students averaged 25.8 out of 60; the summer students averaged 32.9.

By the end of the summer, enrollment had shrunk to 46 (22 in the morning section, 15 in the afternoon, and 9 in the evening). The attrition rate was not significantly different from that of previous summers or the regular academic year.

4 Discussion

Not only was student performance from the summer course high, but students also found the course quite enjoyable. We initially worried that students would not have the patience to work at the computer for three consecutive hours. This worry was unfounded. In fact, it was often difficult to get students to leave. Table 3 includes some quotes from online surveys given during the course. Although there was some variability in responses, the majority of students were positive about their experiences. The majority recommended continuing keeping our experimental format instead of the traditional, lecture-based format. Additionally, the instructor (Clancy) received his highest evaluation scores in twenty-five years of teaching, and the teaching assistant (Ryan) received his highest ratings in six semesters of working with the course. Most complaints in the surveys were with regards to the initial buggy state of our software, which was fixed while the course progressed.

Another promising finding concerns equity: Students who were likely to have performed poorly in a CS 3 course did reasonably well in this summer's offering. Many students commented that the course was quite difficult, but most of these did reasonably well.

It became clear during this course that the role of the instructor had changed fundamentally, compared to traditionally formatted courses. Instead of being primarily a lecturer, the instructor becomes a tutor, spending most of his time engaging the students one-on-one. Cognitive research has shown that tutoring is the most effective way of enhancing student learning. With collaboration and quiz tools that allow the instructor to continuously monitor students' understanding, s/he can more efficiently target the tutoring attention where it is necessary. Consequently, staff training, especially on tutoring strategies for lab assistants, becomes a much more important role for the instructor in courses with this pedagogical format.

Our findings from this summer have many implications, and we have future research plans to investigate many of them. It is clear that technology can enable new forms of interaction between instructor and student, and enhance the learning experience for students. It is also clear that introductory programming courses benefit from this approach, and it is likely that a wide spread of the computer science spectrum can do so as well.

There are clear implications for distance learning, although having students engage concurrently is important. We have plans for extending our pedagogical approach to community colleges and other four-year colleges, with labs running synchronously across remote settings. Future research is necessary: the summer students may be different, qualitatively, than regular semester students. We are running a section of the 2002 fall semester CS 3 at U.C. Berkeley in the same way as the summer course, and plan to run multiple sections in spring 2003.

Finally, our software and pedagogical approach should allow better CS education research, making it easy to modify curricular segments across sections or individuals. The on-line collaboration and quizzing tools allow results to be quickly measured and inspected.

References

[1] Bransford, J. D., Brown, A. L., & Cocking, R. R. (Eds.). (1999). How people learn: Brain, Mind, Experience, and School. Washington, DC: National Research Council.

[2] Clancy, M.J. and Linn, M.C. "Case Studies in the Classroom", proceedings of the 23rd SIGCSE Technical Symposium on Computer Science Education, Kansas City, Missouri, March, 1992; published as *SIGCSE Bulletin*, volume 24, number 1, March 1992.

[3] Clancy, M.J. and Linn, M.C. "Functional Fun", proceedings of the 21st SIGCSE Technical Symposium on Computer Science Education, Washington, D.C., February, 1990; published as *SIGCSE Bulletin*, volume 22, number 1, February 1990.

[4] Davis, E.A., Linn, M.C., and Clancy, M.J. "Learning to Use Parentheses and Quotes in LISP", *Computer Science Education*, volume 6, number 1, 1995, pages 15-31.

[5] Harvey, Brian and Wright, Matt. *Simply Scheme: Introducing Computer Science* (second edition), MIT Press, 1999.

[6] Linn, M.C. and Clancy, M.J. "The Case for Case Studies of Programming Problems", *Communications of the ACM*, volume 35, number 3, pages 121-132, March 1992.

[7] Linn, M.C. & Slotta, J.D. "WISE Science." *Educational Leadership*, 52, 29-32 (2000). Association for Supervision and Curriculum Development. Alexandria, VA.

[8] Mann, L.M., Linn, M.C., and Clancy, M.J. "Can Tracing Tools Contribute to Programming Efficiency? The LISP Evaluation Modeler", *Interactive Learning Environments*, volume 4, number 1, 1994.

[9] Slotta, J.D (in press). "The Web-based Inquiry Science Environment (WISE): Scaffolding teachers to adopt inquiry and technology." To appear in M.C. Linn, P. Bell and E. Davis (editors). *Internet Environments for Science Education*. LEA.

[10] Urban-Lurain, Mark and Weinshank, Donald J. "'I Do and I Understand:' Mastery Model Learning for a Large Non-Major Course", proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education, New Orleans, LA, March, 1999; published as *SIGCSE Bulletin*, volume 31, number 1, pp. 150–154.

day(s)	*topics/activities*	Simply Scheme *readings*
June 24	getting oriented to the UC-WISE system; communicating with the Scheme interpreter	chapters 3 and 4
June 25	working with words and sentences	chapter 5
June 26	working with conditional expressions	chapter 6
June 27	putting conditionals and words and sentences together	
June 28	catch-up day	
July 1	working with the “Difference Between Dates” case study	
July 2	more work with the “Difference Between Dates” case study	
July 3	first programming challenge (online exam)	
July 5	work on the “Difference Between Dates” project	
July 8	starting to think recursively	chapter 11
July 9	working with more complicated recursive procedures	chapters 12, 13
July 10	exam; working with even more complicated recursions	chapter 14
July 11	working with the “Roman Numerals” case study	
July 12	work on the “Number Spelling” project	
July 15-16	more work on the “Number Spelling” project	
July 17	catch-up day	
July 18	generalizing procedures by using procedures as arguments	chapter 8
July 19	more work with higher-order procedures	chapters 7, 9, 19
July 22	second programming challenge	
July 23-24	working with the “Tic-Tac-Toe” program	chapter 10
July 25	working with the pattern matcher	chapter 16
July 26	working with lists	chapter 17
July 29	exam; advanced list processing	chapter 15
July 30	more advanced list processing	
July 31-August 2	the “Doctor” project	
August 5-8	work on final project	
August 9	deadline for project progress checkoff	
August 12	work on final project	
August 13	deadline for project progress checkoff	
August 14	work on final project	
August 15-16	final exam	

Table 1: Day-by-day topic/activity organization.

Type	*Description of Activity*
Quiz	Define a function, then determine the value of an expression involving nested function calls.
Web-text	Introduce the terms “word” and “sentence” and the functions `first` and `butfirst`.
Gated collaboration	Why does `(first mike)` produce an error, while `(first (quote mike))` returns “m”?
Web-text	We note that numbers are self-evaluating.
Gated collaboration	Given functions `(define (initial1 name) (first name))` and `(define (initial2 name) (first (quote name)))` try to use each to find Mike’s first initial by supplying `"mike"` as an argument. Explain the results.
Web-text, exercise	Introduce the `word` procedure, and ask for function `plural` that returns an “s” onto the end of its argument.
Web-text	We describe the usage of the quote mark to abbreviate the `quote` function.
Gated collaboration	Predict the result of several uses of `sentence` and `word`.
Web-text, exercise	Fill in the blanks in a given collection of expressions to get specified results.
Web-text	We explain the difference between using `first` and `butfirst` with sentences and using them with words.
Gated collaboration	Without the Scheme interpreter, evaluate several expressions using `first` and `butfirst` with words and sentences.
Web-text, exercise	Fill in the blanks in a given collection of expressions to get specified results.
Gated collaboration	Explain the difference between `(first 'mezzanine)` and `(first '(mezzanine))`, and between `(first (square 7))` and `(first '(square 7))`.
Web-text, exercise	Supply parentheses and quotes in a sequence of words to result in an expression that evaluates to a given value.
Gated collaboration	Experiment with a built-in function, finding out how many arguments it takes and what it returns.
Discussion	What are good ways to experiment with a function?
Web-text	Outline the Scheme evaluation algorithm.
Gated collaboration	Describe the result of coding plural as `(define (plural word) (word word 's))`
Web-text, exercise	Write and test functions that convert back and forth between an inch count and a feet/inches height measurement.
Web-text, exercise	Write a function that translates a day in the French Revolutionary calendar year to its month/date representation
Homework	Describe what `''banana` stands for; explain what the value of `(first ''banana)` is.
Homework	Write a function that inserts the word “and” as the next-to-last word in its nonempty sentence argument.
Homework	Contribute to online discussion of why parentheses and quotes are so hard for the typical student to understand.

Table 2: Sample activities from the second day of our course. All activities were completed by students using WISE.

I'm enjoying [this course] much more than I expected. I’ve always been told that CS 3 is a boring class, intended for those who only need to take it as a prerequisite or for Business students who need an A in a computing class. I like it because it requires a sort of new approach to thinking.

I think this experimental course should be implemented because I prefer this layout to the lecture, discussion, lab layout immensely and the book goes along very well with the course.

I enjoyed CS 3 very much. It was interesting to have the course taught a different way than in other classes. I felt I had a lot more one-on-one interaction than in my previous classes.

I really enjoyed this class! I think it was well-structured and well-taught. I really think future CS 3 classes should be taught this way.

All criticisms aside, I've enjoyed this system of class *far* more than the regular kind (which, I admit, I didn’t spend a terrible amount of time in)—I mean, it makes me actually *want* to listen to the professor when he occasionally interrupts our work for a small lecture—yes, you heard me, *I want to listen to the professor lecture*!

I really like the format of the class because there is a lot of contact with the professor—he is very helpful and approachable. The course doesn’t have the impersonal ambiguity of a large lecture hall. The structure of the class—spending every day at a computer with daily lessons is good because the lessons show us where we should be by the end of each day yet allow us to work at our own pace and take it home if necessary. Actually working on problems all day is a great way to learn—through experience—as opposed to just reading or hearing about it.

I like the way lectures are combined with problems online. I am learning more about CS than I ever did before. This method of teaching is very practical and works a lot better than regular lecture/lab/discussion sections. … the professor is always there ready to answer any question you may have. People are not afraid to ask the stupid question because it does not interrupt anyone else's progress. I really support this type of teaching.

Table 3: Selected quotes from online surveys given during the course