For consideration at FECS'12

Supporting active learning in large classrooms using pen-enabled computers

Sam Kamin Wade Fagen
Department of Computer Science
University of Illinois
Urbana, IL 61801 USA
{kamin, wfagen2}@illinois.edu

ABSTRACT

Employing active learning methods in a large college class presents a variety of challenges. One of the most important is that it is difficult for the lecturer to properly calibrate the learning activities. We present our experience in such a class – an advanced, required class in Computer Science, with an enrollment of 150 students – where we employed a computer system based on pen-enabled computers to provide real-time feedback to the professor. The system helped to keep the students engaged, while having them engage in realistic, homework-like problems in the classroom. We describe the system and our experience with it.

KEYWORDS

Software Framework, Active Learning Tool, Collaborative Learning Tool

1. INTRODUCTION

The implementation of active learning in large college classes poses special challenges [1]. Some of these can be addressed with technology. Electronic voting systems (``clickers'') are becoming quite popular; these involve students in solving multiple-choice problems in the class, usually collaboratively. This keeps students more engaged in the class --- which in the modern college classroom has become a difficult task for all but the most entertaining instructors. However, in some circumstances, clickers are difficult to employ. In this paper, we describe one such circumstance, and discuss how we employed more advanced technology, with richer feedback, to enhance an active learning course.

The first author teaches a senior-level course in Computer Science at the University of Illinois, CS 421, Programming Languages and Compilers. The course is required for CS majors, and enrolls about 150 students. We wanted to employ an active learning approach, in which a significant portion of the class time would be devoted to student problem-solving. The class exercises should be of the same kind that the students would do for homework and on exams (including programming). Furthermore, we wanted the discussion of these exercises to be driven by the difficulties the students actually experienced. This raised another problem: Courses at this level --- perhaps especially in Computer Science --- are constantly evolving. We have no reliable history telling us what material will be most difficult for the students, or in what way. So, we wanted to assigned realistic, long-form exercises in the class, and *see* what the students were doing. These were not multiple choice questions; clickers would not solve our problem.

The traditional solution is to assign the exercises, and then walk around the class during the few minutes the students were engaged in them, looking over the students' shoulders. However, this is simply impractical in a large lecture hall, for many reasons. It is impossible to

reach students who are not sitting on the aisles without disrupting other students; time constraints preclude reaching any students at the back of the class. The students' natural writing posture makes it difficult to see what they are writing without making them stop their work. More fundamentally, students are often shy about showing their work; having a professor look over their shoulder seriously impedes their ability to do the exercise. And besides, this would lead to a mode of interaction in which the professor offers help to one or maybe two students, rather than obtaining an overall sense of the classroom; this is fine in a small classroom, where every student will eventually get that kind of attention, but in a large class, the professor needs to focus on the class as a whole.

We solved our problem by developing a system that runs on wirelessly-networked, pen-enabled devices, on which some subset of the students solve the exercises (while the rest work on paper). The instructor can view the work of that subset (currently four students, changing constantly during the class), and display it to the class. We developed this system, which runs on Windows-based Tablet PCs [2] and Android devices [3], in our SLICE development framework [4,5], which allows us to develop applications for both platforms at once. We have used the system in every class in the current semester.

The benefits of this approach are:

(1) The exercises are not confined to multiple choice, or even short answer. They are, as we intended, simple versions of the problems students will confront in their homework.
(2) The students answer the questions in the most natural way, by writing out the solutions by hand.[1]
(3) Students have a sense of participating because, when they are using the tablets --- and every students gets the opportunity, because the tablets change hands after every exercise --- their work is often shown to the entire class.
(4) Because the professor is viewing the students' work remotely and anonymously, the intimidation factor is reduced.
(5) As part of showing a student solution to the class, the professor may write additional details or correct the student work. Starting a discussion with the students, based on the work of a peer, reinforces the sense of participation.
(6) Above all, the original goal is achieved: the professor can see exactly what the students are doing and what difficulties they are encountering, by watching them solve actual problems in real time.

In this paper, we describe the class and the computer system we developed in detail, and relate our experiences in the classroom.

2. THE COURSE

CS 421, Programming Languages and Compilers, is a required course in our CS curriculum, normally taken by students in their senior year. It is taught in twice-weekly, 75-minute lectures, in a large lecture hall; enrollment is approximately 150 students. The course material is generally considered quite challenging. Homework is given weekly and usually involves programming.

The goal of each lecture is mainly to teach the students the new skills needed to complete that weeks' homework. Accordingly, the in-class exercises are versions of the same kinds of problems as in the homework, often building up from much simpler versions of

[1] We have heard it said that students no longer write by hand, but instead type; and we have even heard that this especially true of Computer Science students. In our experience, this claim, which is made mainly by non-CS faculty, is completely divorced from reality, as one could see by visiting any CS classroom. Anecdotally, the first author has long banned laptops from his classes; out of hundreds of students, only two have ever claimed that this disrupts their note-taking

those problems, which can be solved completely in the class, to fuller versions that can only be solved partially. We hope to leave students in a good position to complete the homework. For example, the second assignment of the semester is to write functions over linked lists using recursion, in the programming language OCaml [6]. The in-class exercises that week built up from simple *non*-recursive programs on lists, to complicated recursion. Later in the semester, we talk about ``compilation schemes'' – rules for generating machine language code for various high-level language constructs – and the exercises involve writing these rules; we start with simple examples that involve little more than copying examples given by the instructor, to creating rules from scratch.

The exercises are included within the class lecture slides. The slides containing exercises are printed before class and handed out to all students. (The lecture slides themselves are posted online before class, but very few students print those ahead of time.) Examples of exercises, and student answers, are given in section 4.

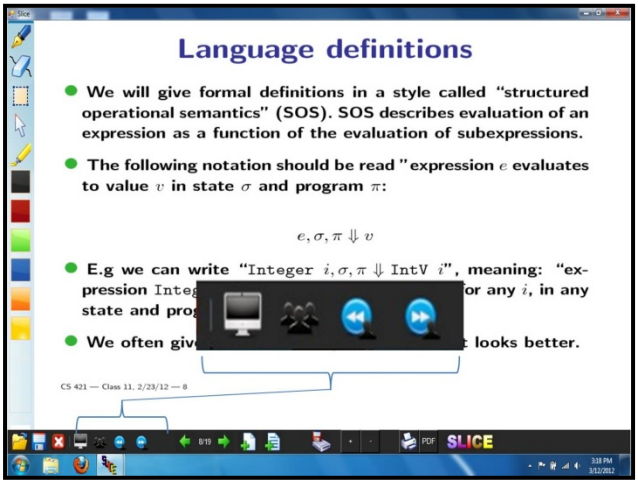

Figure 1: Lecturer's View

3. A MONITORING APP

To allow the instructor to monitor students' work on the in-class exercises, we used our SLICE [4,5] framework to build an application for pen-enabled computers. The application runs on Tablet PCs and on Android devices; in SLICE, the application code is exactly the same for both kinds of devices. The professor has a Tablet PC on which the lecture is given, and can view the work of students on other Tablet PCs and Android tablets.

At the start of each class, we distribute a set of computers (four, at present) to students randomly.[2] All the professor's notes, including the exercise slides, are transmitted to those machines wirelessly. We ask the students who received the tablets to solve the first exercise on them, and then pass them along to someone else. Classes usually have at least half a dozen exercises, so a significant portion of the class will use the tablets at some time during the class.

Figure 1 shows the instructor's version of the app. It is an ordinary Tablet PC-based presentation app, with buttons to change pen colors, erase pen strokes, move to the next slide, and so on. Most interesting is the set of buttons near the bottom left, highlighted in the figure. The ``heads'' button switches the instructor's machine to monitoring mode, displaying the first student's tablet (at the same page as the professor's tablet, that is, the current exercise page). The arrow buttons next to it move from student to students. The display icon on the left displays, on the room display, the page of whatever student the professor is currently viewing; this allows the instructor to either show a good solution or point out an error that he suspects is common. The student's version of the app is similar, but with fewer buttons. It lacks the monitoring buttons, of course, but also lacks the erase button; students can only erase by crossing out. This was intentional, so that the instructor could see the students' false starts.

[2] We chose the number four for the most mundane of reasons: it is the number we can comfortably carry from our lab to the class. The app can handle any number of student machines, although there is a real question of how many response the professor can realistically monitor during the exercise periods.

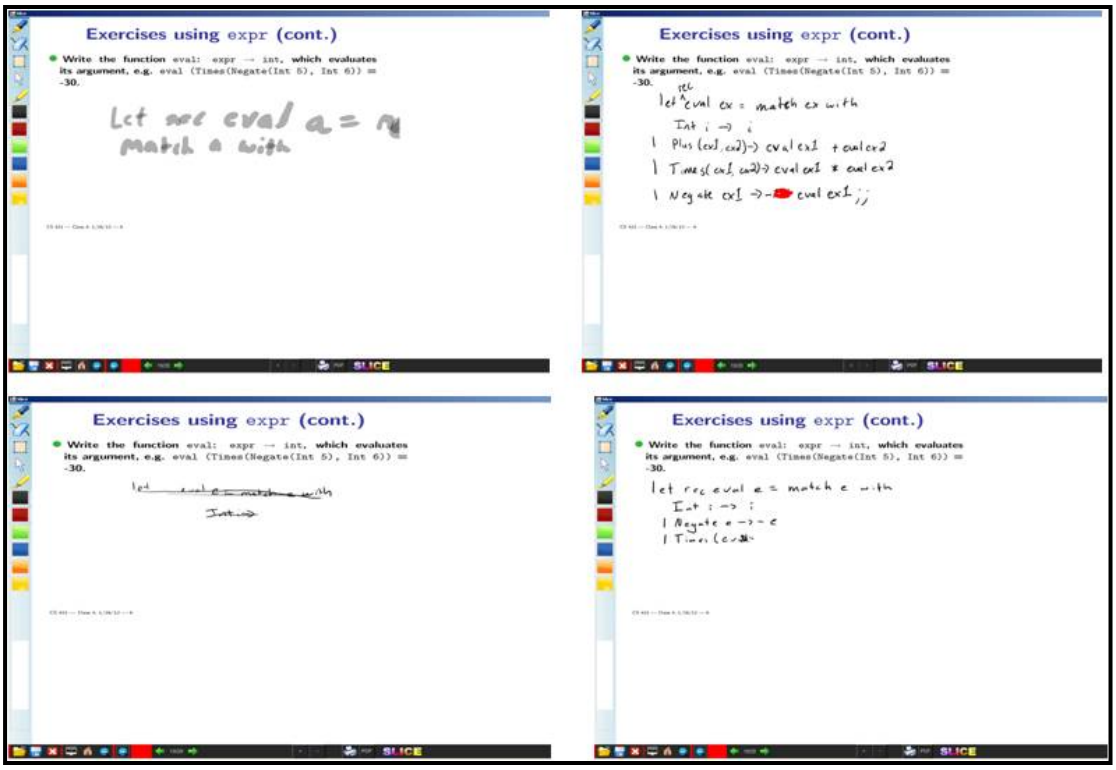

Figure 2: Four students doing the same exercise, viewed by the professor.

4. EXPERIENCE

The application is effective in giving the lecturer a window on the students' thought processes. The students seem to enjoy using it, and will sometimes draw pictures, or write their answers in a rainbow of colors. (The machines are very simple and natural to use; we have had no need to give the students any training on how to use them.)

Sometimes, the best information comes from the *lack* of student responses to an exercise; if none of the four students is working on an exercise after, say, one minute of thinking, that strongly suggests that those students --- and most likely, *almost all* the students --- are confused by the problem. But one of the benefits of getting this feedback constantly is that the professor learns over time how to pace the class, so that he gives fewer and fewer exercises like that. Or, to put it differently, he learns how to be so clear and concrete that students always understand what he asking of them. Nearly all the time, the students respond, although at very different speeds.

More typical are responses like those shown in Figure 2, where most of the students are completing the questions. One student will get the first part of the problem, and the professor will use that as the starting point for a discussion, and so on. (These screenshots show the students' final responses, after they have benefited from the entire discussion.) In programming classes, it is worthwhile to point out even trivial mistakes, because these can cost a lot of time when they sit down to do their homework.

Another way the system supports active learning is that it helps the teacher *calibrate* the pace of the exercises and, by extension, of the course material itself. The best example comes from the very beginning of the course. Lecture 2 covers basic material that the students have seen before: writing recursive functions. Either because they hadn't done this in some time, or because they were using a new programming language, they struggled with these exercises.

Because the material is so fundamental to the course, the professor changed the schedule of the course, and continued working on these problems in class 3. Absent the monitoring system, it may have seemed that the students were confused or disengaged, but the system provided concrete, unignorable evidence that he needed to spend more time on this topic.

Having said that, we should add that, despite the concern voiced in [1], we did not, in the end, cover less material than in previous semesters. We are certain of this because we can compare the exams. What we did was to jettison some "filler" material, leaving only material that represents concrete skills that can be tested. We view this, in itself, as a positive contribution of the monitoring system.

5. CONCLUSIONS

The essential thing in employing active learning is that the activities in which students engage should be *challenging* but *doable*. Careful calibration of the exercises is achievable if either the topic is well-studied and the way that students learn it are well understood, or if there is a tight feedback loop between students and instructor in the classroom. Neither of these conditions obtain in large college classes.

We have presented a system in which an instructor can closely monitor the work of a subset of the students in a class, and thereby create a facsimile of this tight feedback loop. This system uses wirelessly-connected, pen-enabled computers that are distributed to students randomly at the start of class, and move around during the class. The instructor, employing a networked tablet computer, can switch from lecturing mode to monitoring mode at will. When the students might benefit from seeing the work of another student, the instructor can display it on the room display. The system was developed using the SLICE framework, which greatly simplified development of applications like this one.

In the end, perhaps the greatest benefit of the system is that the professor gets a very clear picture of what the students are getting from the lecture. It forces the professor to focus less on *what he or she is teaching* and more on *what the students are learning*. And that is good for everyone.

REFERENCES

[1] Charles Bonwell and James Eison. Active Learning: Creating Excitement in the Classroom (J-B ASHE Higher Education Report Series (AEHE)). Jossey-Bass. 1991.

[2] Microsoft Tablet PCs, Wikipedia entry: wikipedia.org/wiki/Microsoft_Tablet_PC

[3] Android, Wikipedia entry: wikipedia.org/wiki/Android_(operating_system)

[4] S. Kamin, M. Hines, C. Peiper and B. Capitanu. "A System for Developing Tablet PC Applications for Education." in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. 2008.

[5] Wade Fagen, Sam Kamin. "A Cross-Platform Framework for Educational Application Development on Phones, Tablets, and Tablet PCs. *submitted for publication*. (Downloadable at slice.cs.illinois.edu.)

[6] OCaml website: caml.inria.fr/ocaml/