

# Large Introductory Computer Science Classes: Strategies for Effective Course Management

David G. Kay

Department of Information and Computer Science

University of California

Irvine, CA 92697–3425

(714) 824-5072

kay@uci.edu

**Key Words:** large courses, introductory courses, course management, teaching techniques

## Abstract

Traditionally, a large introductory course meant a lecture hall with a single lecturer talking and students taking notes—but no longer. Today a wide variety of techniques, not only in the classroom but in labs and faculty offices and cyberspace, can make a large introductory course an extremely effective educational experience. We explore these practices, with pointers for further information, as a guide for instructors or departments faced with the large-course model of instruction.

## THE CHALLENGES OF LARGE CLASSES

Introductory computer science courses present their own unique challenges. The range of student backgrounds may be very wide in a single course, from complete novices to students with many years of self-taught experience. The specification of programming assignments, like the specification of any software, is notoriously difficult. Assignments, particularly those involving production of software, are voluminous and require painstaking evaluation. The time demands on students are greater than in non-laboratory courses, and they increase further as more open-ended projects are assigned.

Couple this with an enrollment of 100 students or more (in some institutions, enrollments of 500 are not unheard of), and the challenges multiply.

We will characterize these issues below, and go on to consider practices and strategies that may help resolve these difficulties and occasionally even turn them to advantage.

## Increased Size, Increased Workload

Outside of the classroom itself, the work of grading and individual assistance increases at least linearly with enrollment. Also, if the students in a course are distributed normally, the distance between the strongest and the weakest will probably be much greater than in a small class, which means that the students at either end of the scale are likely to require even more extra attention.

Even the effort of preparing materials, assignments, and exams is greater in a large course, with more concentrated advance preparation. Examination questions that can be graded rapidly (of which multiple-choice questions are an extreme example) typically require more preparation time than open-ended questions, but as enrollments increase, the value of that tradeoff changes. Likewise, extra time spent perfecting the specifications of lab assignments becomes essential with higher enrollments, since fielding individual questions to clarify ambiguities or errors from, say, 20% of the students becomes unmanageable as the class size increases. This added preparation time is a significant extra burden in teaching large courses.

## Student Anonymity

In large classes, individual students can remain largely anonymous. This can result in inattention—think of students in the back of the room reading the newspaper or, more currently, playing solitaire on their laptop computers. Inattention can produce apathy and disconnectedness—when the teacher takes an active personal interest in a student's progress, the student works harder and learns better, but personal contact is limited in a large class. The ultimate result may be attrition—when students are not engaged and their performance suffers, they find it easy simply to slip away.

## Distributing the Work Yields Inconsistency

Distributed processing addresses many of these problems, as we divide students into smaller discussion and lab sections and apportion their work to assistants for grading. But this leads to yet another problem: inconsistency in what the students learn and in the grades they receive.

## TECHNIQUES AND STRATEGIES

Below we catalog many of the strategies successful instructors have used for resolving these issues.

### Course Organization

As mentioned above, the classic organizational approach to accommodating the increased workload in a large course is to create discussion and/or lab sections, which might range in size from 5 to 30 students. Later we describe strategies for dealing with the potential for inconsistency.

One alternative organization is self-paced courses (as at the University of California, Berkeley<sup>[1]</sup>), in which students learn independently, on their own schedule, making an appointment with a grader as they complete each assignment.

Another approach is to eliminate the large lecture entirely, conducting each section independently. CMU follows this approach, in which the only unifying element is a common examination at the end of the term. [2-4]

Many institutions offer a variety of introductory courses, some more gentle and some more accelerated than the mainstream course. To eliminate some attrition, students might be permitted to transfer from one course to another more flexibly—perhaps at a later point in the term—than is the norm.

Even in a single course, some variation in assignments can accommodate a wider range of students. Contests or extra credit can give the strongest students an opportunity to shine. [5]

### In the Classroom

A variety of active learning techniques can keep students engaged with the material in class. At intervals during the lecture, the instructor can ask the students to spend a few minutes working out a problem, either individually or in small groups. [6,7]

Passing out overhead transparencies and pens allows some students to share their work, providing valuable discussion material for the whole class.

With the greater availability of in-class computers and projection equipment, real-time demonstrations become more feasible. While an entire class period in a darkened room may tax any student's ability to stay awake, judicious use of these techniques can be highly effective at maintaining interest. [6]

As size increases, the number of students who must unavoidably miss class increases as well. Audio- or videotaping lectures and making the tapes available for

later review provides students an easy way to catch up, and one more complete and effective than just relying on a classmate's notes. In general, fostering a sense of community in the course helps keep students engaged. Eric Roberts points out that the energy and momentum of a large class may provide an advantage over small classes. [5] Community can be fostered by running jokes, by class parties, by a course web page or on-line discussion group.

### Out of the Classroom

Student alienation decreases when the instructor is accessible outside of class. The solitary lecturer appears distant and unapproachable; having frequent office hours and repeatedly soliciting attendance helps break the barrier down. At UC Irvine, the campus subsidizes a "take-a-professor-to-lunch" program in which a group of five students may invite a professor to lunch in any campus food facility; each student receives a one-dollar discount and the faculty member gets a free lunch, above and beyond the opportunity for interaction on a human scale.

Learning students' names is another tried-and-true technique. Learning over 100 names may not be feasible for every instructor, but these techniques make it possible to learn a significant proportion: Require each student to provide a snapshot (with the name on the back) and review them like flash cards, or have every student give his or her name before asking in-class questions.

Electronic mail (sent privately and individually to the instructor or assistants) and on-line bulletin boards or news groups (with postings accessible to the entire class) can be channels for class announcements, questions, and discussion, greatly enhancing availability and creating community.

One flexible approach to managing the volume of messages is to have a course mailing address that goes to the entire teaching staff. Then, whoever reads the mail first can reply to it, with a copy back to the list so that the others will know a response has been sent. This provides rapid turnaround for the students, distributes the correspondence workload, and gives the instructor a real-time picture of how the course is going. Any messages of general interest to the class can be posted or forwarded to the group as a whole (perhaps with identifying information removed).

Web-based software [8] can maintain archives of official announcements and can, by coordination with the registrar, maintain electronic class mailing lists that reflect up-to-date enrollments. Of course the Web and other technologies can help with a variety of course-management tasks. [7,9,10]

### Assignments and Grading

Even when courses are divided into sections, often the

students take a common exam. To preserve consistency across sections when grading the exam, it is best for a single individual to grade every student's response to a given question. Where the enrollment is too large to permit this, great care must be taken to design formal rubrics or grading standards and to ensure adherence to it by cross-checking graded problems periodically during the process. [11] This is time well spent, because students do compare their results and will focus laser-like on any perceived inconsistency.

Programming assignments are no exception to the general difficulty of specifying software requirements. Questions about an assignment's requirements always arise, and teaching staff may quite reasonably give contradictory answers. Expecting to eliminate all ambiguity, at least in larger, design-oriented assignments, is unrealistic.

One approach to this issue is to let each section leader make these small decisions for his or her section, assign grades accordingly, and perhaps recognize the lack of comparability among sections when assigning final grades, as described below.

A second approach is to identify for each assignment a single person with decision-making authority, typically one of the section leaders. Other section leaders will defer all substantive questions to this authority, who will issue consistent answers and clarifications (electronic mail or bulletin boards help immensely here). This authority can also take the lead in grading that assignment consistently with his or her pronouncements, by preparing the scoring standards if not by doing all the grading. Of course this decision-making position could rotate among the teaching staff from one assignment to the next.

When setting grades, the conventional approach is to grade the entire class on one scale (be it curved or fixed). In a typical course, lab assignments might count for 40% of the course grade and exams might count 60%. The exams might well be graded comparably across sections, as described above. The assignments, however, would more typically be graded by a different person for each section. Those scores might also be comparable across sections, perhaps by following the practices described above, but often they are not. How, then, can we assign course grades consistently?

One approach to smoothing out these inconsistencies is to normalize the scores across sections, raising the scores for sections that appear to have harsher grading. However, this approach presumes that the populations of each section are comparable, which is probably not realistic. Another approach is simply to recognize the lack of comparability, treating each section as an independently graded course which happens to have 60% of its grading in common with the other sections. To follow this approach, the instructor must explain to students from the outset that grades are not comparable across sections because each grader grades

differently; the instructor must also meet with each section leader to set the grades, perhaps using the common 60% from the exam scores to constrain the variation among sections.

## Staff Management

A major part of the workload in teaching large courses is managing the teaching staff—teaching assistants, section leaders, lab assistants, tutors, graders, and so on. Recruiting graduate students as teaching assistants is harder in computer science than in many disciplines, because computer science students have opportunities for research assistantships or outside employment that are often not available to students in other areas. One often-overlooked source of teaching staff is advanced undergraduate students; with appropriate selection and training, they can be extremely effective section leaders, tutors, and graders. Moreover, often they are paid at a lower rate than graduate students, providing more staffing for the same price. [7,12]

Some people may be born with teaching talent, but every section leader should be exposed to departmental policies and the issues surrounding teaching undergraduates in the university (including respect for diverse backgrounds); nor should their first experience speaking before a group or grading students' work be in the section they lead. Prospective or first-time section leaders should have some kind of training as described in [13].

Above and beyond training, the teaching staff of a large course should meet periodically to compare notes, coordinate policies, discuss current difficulties, and exchange experiences. Staff meetings are an important way for the instructor to get feedback about the progress of the course from the people on the front lines. In-person meetings work better than electronic mail exchanges alone, because they provide a specific time to focus on the course.

## In the Lab

Having lab partners, group work, and peer testing of students' programs can all diminish feelings of alienation, as can face-to-face grading.

Electronic submission of programs and automated testing of correctness can not only make the grading workload more manageable, but also provide a more thorough assessment than traditional grading and free the grader's time to consider issues such as design, style, interface, and documentation, which are not amenable to automated assessment. [14-19]

## Viewing the Future

With the advent of the World-Wide Web and an increased focus on distance education, one might expect

introductory courses to be larger still in the future. Yet such classes, typically asynchronous as well as geographically distributed, would have less cohesion and sense of community than a traditional course. The information bandwidth for electronically mediated courses—even with live, real-time video—is still much narrower than for in-person contact. “The next best thing to being there” is indeed second-best; educators estimate that an instructor’s words account for less than half the information conveyed in a classroom. Even in a large lecture hall, the fine points of an instructor’s intonation, gestures, and body language, as well as the reactions and rustling of the audience, add information to the presentation that electronic means typically don’t capture. While technology can improve large courses in many ways, in-person instruction is unlikely to disappear, certainly not without a significant diminution of pedagogical effectiveness.

## Conclusion

We have described a wide range of ways to enhance the quality of large introductory courses. We also note that many departments with large introductory enrollments take a broader approach to this issue by recruiting and retaining permanent faculty as introductory education specialists. [4]

Conventional wisdom holds that small classes are better, but even if we accept that, we cannot simply wish our large enrollments away. The quality of large introductory courses can be excellent with adequate resources and informed, thoughtful management.

## Acknowledgements

Many of the practices in this paper were collected during a panel at the 1997 SIGCSE Technical Symposium [2].

## References

- [1] See <http://www.eecs.berkeley.edu/~selfpace>
- [2] Kay, D. G., Carrasquel, J., Clancy, M. J., Roberts, E., and Zachary, J. Managing large introductory courses (panel presentation). In *Proceedings of the 28th SIGCSE Technical Symposium*, 1997, p. 386.
- [3] Zachary, J. L. Tutorial-based teaching of introductory programming classes. In *Proceedings of the 25th SIGCSE Technical Symposium*, 1994, p. 136.
- [4] Kay, D. G., Carrasquel, J., Clancy, M. J., Roberts, E., and Zachary, J. Large introductory courses in research computer science departments (panel presentation). In *Proceedings of the 29th SIGCSE Technical Symposium*, 1998.
- [5] Roberts, E. Encouraging top students in large introductory classes. In *Speaking of Teaching*, Stan-

- ford Center for Teaching and Learning, vol. 8, no. 2, Winter 1997.
- [6] Rodger, S. H. An interactive lecture approach to teaching computer science. In *Proceedings of the 26th SIGCSE Technical Symposium*, 1995, p. 278.
- [7] Wills, C., Finkel, D., Gennert, M. A., and Ward, M. O. Peer learning in an introductory computer science course. In *Proceedings of the 25th SIGCSE Technical Symposium*, 1994, p. 309.
- [8] See, e.g., The UCI Electronic Educational Environment (EEE), <http://eee.uci.edu>
- [9] Reek, K. A. A software infrastructure to support introductory computer science courses. In *Proceedings of the 27th SIGCSE Technical Symposium*, 1996, p. 125.
- [10] Nishida, T., Saitoh, A., Tsujino, Y., and Tokura, N. Lecture supporting system by using Email and WWW. In *Proceedings of the 27th SIGCSE Technical Symposium*, 1996, p. 280.
- [11] Faster, fairer, and more consistent grading using techniques from the advanced placement reading (panel presentation). In *Proceedings of the 21st SIGCSE Technical Symposium*, 1990, p. 266.
- [12] Roberts, E., Lilly, J., and Rollins, B. Using undergraduate teaching assistants in introductory programming courses: an update on the Stanford experience. In *Proceedings of the 26th SIGCSE Technical Symposium*, 1995, p. 48.
- [13] Kay, D. G. Training computer science teaching assistants: a seminar for new TAs. In *Proceedings of the 26th SIGCSE Technical Symposium*, 1995, p. 53.
- [14] Kay, D. G., Isaacson, P. C., Scott, T. A., and Reek, K. A. Automated grading assistance for student programs (panel presentation). In *Proceedings of the 25th SIGCSE Technical Symposium*, 1994, p. 381.
- [15] Kay, D. G. User environments for student programmers. In *The Role of Programming in Teaching Informatics*, M. Griffiths and D. Tagg, Eds. North-Holland, 1985.
- [16] Burris, H., and Darr, M. The PROGRAMS Package for Integrated Grading. Program in Computing, Department of Mathematics, University of California, Los Angeles, 1988.
- [17] Reek, K. A. The TRY system, or how to avoid testing student programs. *SIGCSE Bulletin* vol. 21, no. 1, 1989, p. 112.
- [18] Isaacson, P. C., and Scott, T. A. Automating the execution of student programs, *SIGCSE Bulletin* vol. 21, no. 2, 1989, p. 15.
- [19] Jackson, D. and Usher, M. Grading student programs using ASSYST. In *Proceedings of the 28th SIGCSE Technical Symposium*, 1997, p. 355.