Teaching

Andrew P. Black

November 10, 2004

1 My Teaching Agenda

I like to teach — when all else is said and done, that is why I am a Professor rather than a research scientist. I like to have some effect on the world: however trite it may sound, it is important to me to feel that something that I have done has made a difference. If I am very smart and very lucky, over the course of my professional career perhaps a handful of my research ideas may make a difference to the next generation of computer scientists. But over the same time I may be able to influence the lives of hundreds or even thousands of students.

Thus, the question that driven my teaching has been: how can I be most effective at influencing students? What topics should I choose to teach, and how can I teach them most effectively?

I believe that the mission of a university is education rather than training, and that when our students graduate they should *know*, or, if we have been peculiarly successful, *understand*. How can this mission be implemented in a field like Computer Science, where our stakeholders in industry are telling us that our graduates must be able to *do*, in other words, that they want us to teach skills rather than concepts?

I think that the contradiction is more apparent than substantive. The real stakeholders of our University are the students, not their prospective employers. It is our duty to give those students something that will be of value not only on the day that they graduate, but five, ten and even fifty years later. I do not believe that *any* of the computer skills that we teach today will have any value in fifty years, and few will have much value in ten. But the concepts and principles are timeless. In fifty years uncomputable will still be uncomputable, the well-written paper will still be enjoyable, the quadratic algorithm will still be slower than the logarithmic algorithm, and it will still be impossible to know the reason that a remote network host has not responded.

That said, there remains the question of *how* we teach such fundamental concepts. The answer, of course, is "by example": most people are much better at generalizing from a series of examples of increasing complexity than they are at grasping an abstract concept presented in all of its generality. The appropriate use of examples is, I believe, the key to satisfying both the short term needs of the students (and their prospective employers) for immediately usable skills, and the long term needs of the students (and of their prospective employers) for a conceptual understanding of the core

ideas that make computing a science. *When we choose our examples, we should do so by looking forward to the next technology rather than by looking backwards to the last.*

It isn't easy to do this. It means that we must constantly be revising our course materials and our teaching methods. I first taught Object-oriented Programming in 1994–95, when there was a need for a course on this topic at OGI. I have taught this course almost every year since, both alone and in collaboration with other instructors. It has never been the same twice. I have varied the language technology, starting by teaching myself Smalltalk because that was the hot language in 1994, switching into Java when it became available, combining Self and Java when it became clear that many important concepts could not be expressed in Java, moving from Self back to Smalltalk when I found that Self was impractical, and, most recently, in Fall quarter 2004 at PSU, focussing wholly on Smalltalk again because the students had already acquired Java syntax in other classes, and would benefit most from an intensive focus on the concepts of objects-orientation. Other aspects of the course have changed too. In 2004 I think it essential to talk about test-first programming, pair programming, patterns, and refactoring; these ideas were hardly known ten years ago.

2 Teaching Techniques

I started teaching using overhead projector slides, because that seemed like the only practical way of conveying Pascal syntax to a lecture hall containing 50 students, and because it gave me a crutch to lean on when I suffered from stage-fright. I occasionally still use slides, now projected from my computer. Slides are a wonderful way of sharing a diagram or figure with the class, which we can then discuss, critique and argue about. They are also a wonderful way of getting text from my computer to the student's ring binders without it ever passing through their heads. I don't like slides full of text for teaching a class.

Now, I find that it is more effective to talk to the students, to lecture at the whiteboard, and to prompt them to ask questions, and sometimes to answer the questions themselves. I use anecdotes and personal stories to try and convince the students that what we are discussing is relevant to them. Getting the students to work in small groups can help them start asking questions of each other, which then leads to a greater willingness to ask questions of the instructor. In advanced graduate courses I have had a lot of success with assigning readings ahead of class, requiring a written summary of the reading to be brought to class, and using the class period to clarify the major points of the reading, often using a diagram from the assigned text as a talking point. I have found that after the students have come to terms with the idea that they actually have to be awake in class and that they won't have a set of forty slides to take away with them, but must actually take notes and decide for themselves what is noteworthy, the students learn more, enjoy class more, and rank the class more highly when I teach in this way than when I lecture from prepared material.

3 Teaching Record

At the University of Washington, I was a research faculty member, and classroom teaching was not part of my responsibilities. Nevertheless, I volunteered to teach one course per year. I Unfortunately, no longer have student evaluations for these classes.

1981–2	Introductory Programming
1982–3	Formal Semantics
1983–4	Computer Reliability and Nuclear War
1984–5	A Rigorous Approach to Programming
1985–6	Concepts of Programming Languages

In 1986 I taught, as a consultant, a course on Object-oriented programming to professional engineers at an instrument manufacturing company. There were no formal evaluations, but the manager who hired me was sufficiently pleased that he paid me more than the amount for which I had contracted.

In the Fall of 1991, while working at Digital, I was asked to teach the Operating Systems course at Harvard. As a result I became involved in organizing a reading group on types in object-oriented programming languages, in collaboration with Prof. Peter Wegner, who was at that time on sabbatical at Harvard. I presented a tutorial on types at the 1992 OOPSLA conference; there were over 70 attendees. Several years later, I was still receiving requests for letters of recommendation from the students who took that class. Somewhat surprised, I asked one student why he had asked me for a letter: why not ask one of his "real" Harvard professors, who would in any case have more recent knowledge of the student. His answer surprised me. He said that he felt that none of his full-time Harvard professors knew who he was, whereas I had taken an interest in his work. Moreover, he said that it was my class that made him to decide to apply for graduate school in computer science. Experiences like this helped me realize that I should leave industrial research and return to academia.

In Spring 1995, shortly after arriving at OGI, I developed and taught a new graduate course in Object-Oriented Programming. The course combined traditional OOP using Smalltalk with recent research in Distributed Object-Oriented and Typed Object-Oriented Programming. I revised this course in Spring 1996, and included Java as well as Smalltalk. However, the students liked the revised version less well. I decided that this was because the mix of conceptual and pragmatic material appealed to two different audiences, and in 1996-7 split the course into two: a more pragmatic course co-taught with Prof. Dylan McNamee, and a more conceptual course taught with Prof. Ewan Tempero. The conceptual course was well received by the two students that enrolled!

In Winter 1998 I tried again to create a combination of the practical and conceptual. We required OO Analysis and Design as a pre-requisite, dropped Smalltalk in favour of Self, kept Java, and spent about half of the course on programming language issues and half on conceptual issues. In addition, since PSU was at concurrently offering two courses that focussed on the pragmatics of programming, we felt that the students who enrolled at OGI were more likely to be interested in the conceptual material. This course was a success, and I used variations on this format in 1999 and 2000.

In Winter 1998, Prof. John Launchbury and I combined forces to teach the Scholarship Skills course that had been pioneered by Prof. David Maier. We drew heavily on Maier's materials, while adding some innovations of our own, most significantly, a mock programme committee for an imaginary conference. I felt that a course of this type was an important addition to our curriculum, but the experience of teaching it had made it clear that many of our students need remedial English classes before they were ready for Scholarship skills. I worked with the OGI administration to create such a class and make it available across the campus.

In academic year 1999-2000 I took a break from teaching Object-Oriented programming, and designed and taught a new course in Distributed Systems. I taught this course again in Winter 2001 and 2002, with the addition of more practical material and a collaborative project. In 2004 I co-taught Distributed Systems with Prof. Jonathan Walpole.

In Spring 2001 I organized a laboratory-style course on Extreme Programming, but the course achieved insufficient enrollment and was cancelled.

In 1996 I organized a reading group in Types and Objects; it achieved a high level of interest at first, but eventually petered out.

In 2002-3 I revived the advanced operating systems course, which had not been taught for some years. This course achieved an enrollment of 7 and very high ratings from the students who took it (3.7/4). Disappointingly, the enrollment the following year was only 3.

In Spring 2003, I once again taught Object-oriented programming, which had been taught by adjuncts in the interim.

In 2004-5 I had planned to take on another new course, Introduction to Automata. Prof. Tim Sheard and I had discussed teaching this in a new way, by replacing the some of the theorem proving with computerized model building. Thus, for example, rather than presenting finite state automata as mathematical abstractions (a 4-tuple consisting of an alphabet, a set of states, a transition function and a start state), we wanted to experiment with presenting them as programs (which would contain data structures corresponding to the same 4-tuple). Then the theorem showing that a non-deterministic FSA could be reduced to a deterministic FSA would be proved by writing a program that converted a non-deterministic FSA *program* to a deterministic FSA *program*. This course was one of a number that Tim and I had proposed to the National Science Foundation as part of a new graduate curriculum emphasizing programs as objects that can be manipulated by other programs. Although the NSF did not fund the proposal, and the disintegration of the CSE department at OGI meant that the new version of the Automata course was never designed or taught, I am still interested in these ideas and would like to pursue them in the future.

Professors Tim Sheard, Sergio Antoy, and myself asked the dean's office to be allowed to submit a curriculum development proposal to the Intel grant competition this quarter, but were told that permission was not granted. I plan to look for other funding sources for this proposal.

Overall, I am proud of my classroom teaching, and see this as an important part of my job as a Professor. I welcome the opportunity to broaden my teaching portfolio by tackling new courses, provided that there is adequate time to plan and prepare the course, and I look forward to the

challenges of teaching at PSU.

I am presently teaching Object-oriented programming once again, this time to a mixed class of Seniors and graduates at PSU. This class is turning out very differently from the nominally similar class that I taught at OGI in the Spring. The students at PSU have much less background, and so I have had to drastically alter the rate at which I cover material, and I am doing a lot of on-the-fly course redesign. However, the students do seem to be both intelligent and ready to learn, and I have been impressed by how much improvement I have seen in their work between the first and third assignments. By the end of the quarter I will have covered far fewer topics in the current class, but I believe that the students will have thoroughly mastered them.

Courses Taught at OGI

Date	Course	Co-taught with:	Enrollment	Student rating
Spring 1995	CSE 509 Object Oriented Programming		38	
Spring 1996	CSE 509 Object Oriented Programming		35	2.7/4
Fall 1996	CSE 509 Object Oriented Programming	McNamee	25	3.0/4
Spring 1997	CSE 529 Object Oriented Principles	Tempero	2	3.4/4
Winter 1998	CSE 509 Object Oriented Programming	Shulman	21	3.7/4
Winter 1998	CSE 569 Scholarship Skills	Launchbury	15	3.4/4
Winter 1999	CSE 509 Object Oriented Programming		16	3.3/4
Fall 1999	CSE 504 Object Oriented Analysis & Design	Delcambre	33	2.9/4
Winter 2000	CSE 509 Object Oriented Programming	Delcambre	29	3.1/4
Spring 2000	CSE 515 Distributed Computing		11	2.9/4
Winter 2001	CSE 515 Distributed Computing		20	3.0/4
Winter 2002	CSE 515 Distributed Computing		13	3.6/4
Fall 2002	CSE 515 Advanced Operating Systems		7	3.7/4
Winter 2003	CSE 569 Scholarship skills	Maier	11	3.4/4
Spring 2003	CSE 509 Object-Oriented Programming		15	3.5/4
Fall 2003	CSE 515 Advanced Operating Systems		3	4.3/5
Winter 2004	CSE 515 Distributed Systems	Walpole	18	4.2/5
Spring 2004	CSE 529 Object Oriented Programming		13	4.0/5

I have attached the student teaching reviews from the 2003–4 academic year to this narrative. These are neither my best nor my worst reviews, but provide a recent sample from three different courses. The attentive reader will notice that in Spring 2004 (things were rather busy at OGI that quarter) I left too much of the grading to the teaching assistant, without providing adequate oversight. Nevertheless, one student still felt fit to write "Dr. Black is in the top three instructors at this school. I would be glad to take a class from him no matter what it was!"

4 Student Supervision

Listed below are the students with whom I have had close contact over the years. I regard them all as successes. Most certainly, they did not all attain Ph.Ds, and indeed, quite a few who initially started working on a Ph.D. never completed. But all of them, I think, benefited from my mentorship. This is true even of a student whom I advised to leave the Ph.D. program after he had struggled for several quarters without making progress on a research assignment. He was naturally distraught at the time that I gave him this advice, yet a year later, having graduated with an MS by coursework, he was seeking me out at lunch to tell me about his job with a local computer company, and thank me for my support while he as at OGI.

I did not personally graduate any Ph.D.s while at OGI. Indeed, I have never really concerned myself with being listed as primary advisor for a good student: working with the student herself was its own reward. During the time that I was department head at OGI, I did not take on any research students, both because I felt that I could not devote enough time to their supervision, and because I did not have any personal funding that could support them, and felt that it would be inappropriate to burden the department budget with their support. However, I did collaborate with several students who were supervised by other faculty; my publication list shows joint work with Tito Autrey, Jon Inouye, Miguel Mira da Silva, Lakshmi Kethana, Rainer Koster, and Ke Zhang.

Once I had decided to step down as department head, I took on a Ph.D. student, a somewhat shy but very quick woman from China, Jie Huang. I worked with Jie for three years, producing several joint papers, and advising her while she passed the Research Proficiency Exam. Her work on Infopipes led her to study video streaming, which she decided was going to be the topic of her thesis. I was both disappointed and proud when Jie told me that she had decided to work with Jon Walpole on Region of Interest in video for her Ph.D.—disappointed because I had hoped to continue working with her myself, and proud because she had gained the independence and self-confidence to make such an decision.

I am currently working with 3 Ph.D. students at PSU. Phil Quitslund had first been an intern for me at OGI while completing his M.S. at PSU. It was the experience of working with me in my lab that made him decide to apply for a Ph.D. Emerson-Murphy Hill was also an intern in my lab, in his case at the end of his junior year at TESC; the internship experience has a similar effect on him. Chuan-Kai Lin came to OGI in Fall 2003, and worked with three or four different professors during his first year, on a variety of projects. He started working with me on Infopipes in September 2004. In addition, I collaborate regularly with Nathanael Schärli, a Ph.D. student of Oscar Nierstrasz at the University of Bern, with whom I have recently published five significant papers; I attach a letter from Nathanael that describes our collaboration.

In case there should be any lingering doubts about my abilities as an advisor and mentor, I am also attaching two other letters of support from former students. Dr. Eric Jul, now a Professor at DIKU in Copenhagen, was one of my earliest students at the University of Washington. Dr. Paul McKenney, a Distinguished Engineer with IBM, was a part-time PhD student who graduated this Summer.

Current Ph.D. Students

Philip Quitslund, Adapting Traits to Java. 2003–present. This collaboration has already resulted in multiple publications related to Multiview, as well as a successful Research Proficiency Exam.
Emerson Murphy-Hill. Fall 2004–present
Chuan-kai Lin. Fall 2004–present

Prior Ph.D. Students

Rebekah Leslie, *A domain-specific language for Infopipes*, Spring 2004
Jie Huang, Infopipes: *An Abstraction for Real-Rate Information Flow*. 1999-2002. This collaboration resulted in multiple publications related to Infopipes.
Lyle Kopnicky: *An AST for Refactoring*
Paul McKenney, *Theoretical Foundations for Infopipes*. 2001–2003.
Joshua Bower-Cooley, *An Infopipe Infrastructure for CORIE*. 2002–2003.

Student Interns

Emerson Murphy-Hill, Undergraduate at The Evergreen State University, Summer 2003 and Summer 2004. The first summer's work resulted in a paper at the practitioners track at OOPSLA'04, as well as Emerson's application to the Ph.D. program at OGI.
Philip Quitslund, Portland State University (Winter/Spring 2003).
Stephen Drew, Undergraduate from University of Waterloo. (Winter 2002).
D. Carlton, Harvard College, 1993.
M. Immel, Harvard College, 1992. This collaboration resulted in a paper at ECOOP'03.

Ph.D. Graduates

Leif S. Nielsen, *Separation of Data Manipulation and Control: A Structuring tool for Concurrent Program Development*, August 1986.
Norman C. Hutchinson, *Language Design for Distributed Applications*, 1987.
Eric Jul, *Object Mobility in a Distributed Object-Oriented System*, 1988. (Prof. Hank Levy was the advisor of record for both Hutchinson and Jul, as I had left the University of Washington before their theses were submitted.)

M.S. Theses

L. Vaitzblit, *A High-Bandwidth, Multimedia File Server*, MIT, 1991 (I supervised Vaitzblit while I was at Digital; Prof. David Tennenhouse at MIT was the advisor of record).
C. Burris, *A Unified treatment of File Version Compaction*, UW, 1987. This collaboration was summarized in a paper at the IEEE Data Engineering Conference.
T. Yap, *Concurrent Euclid Message Module*, UW, 1986
F. S. Hsu, *Re-implementing Remote Procedure Call*, UW, 1985.

M.S. Projects

I. Domenech, The Eden Command Language Interpreter, (1984).
P. Jensen, The Eden Command Language Virtual Machine, (1984).
B. C. McCord, Implementation of the Eden Programming Language, (1984).

Research Assistants (not otherwise included above)

These were all students at the University of Washington during the period 1981-86.

J. Brower (Eden input/Output Module and Eden debugger).
C. Binding (Window-oriented Terminal Handler).
P. Ma (Evaluation of Eden Checkpoint mechanism and construction of CE run-time kernel).

Senior Year (Undergraduate) Projects

These were also students at the University of Washington, 1981-86.

H. Matsuaka (Translator for the 3R programming language).
S. Evans (Debugging, extension and major performance enhancement of the Zed editor).