Teaching Philosophy

Anne Mulhern

December 19, 2009

I have arrived at my teaching philosophy through extensive teaching experience. As a teaching assistant at the University of Wisconsin-Madison I spent three semesters teaching introductory programming; later I was the instructor for an introductory programming language course at Sarah Lawrence College. In the summer of 2005 I taught a short course in the fundamentals of computer science to gifted middle and high-school students. In 2008 and 2009 I elected to teach a seminar on automated theorem proving and mechanizing metatheory using the Coq proof assistant.

In every course I teach I adhere to the following pedagogical principles:

- Principles are more important than details.
- Active experimentation is more valuable than passive absorption.
- Mistakes, crashes and incorrect results are to be understood and learned from.

This semester I am employed as the instructor for the University of Wisconsin-Madison's introductory compiler course. In this course I have introduced a number of innovations which demonstrate my commitment to the principles above.

Immediate Programmatic Feedback In every assignment I include a mechanism for the students to receive programmatic feedback on the quality of their solutions before they submit them.

In the sections of the course that involve the study of context-free grammars and regular expressions I have made extensive use of JFLAP. This is a software application for defining and testing context-free grammars, finite automata, and regular expressions. JFLAP facilitates experimentation with grammars and automata; students can rapidly sketch an automaton and automatically execute it on an input string. Because students expend less effort on the tedious details of executing automata by hand they are able to focus on the theoretically significant and practically important aspects of automata and grammars.

I provide the students with a testing infrastructure for every assignment which requires them to build some compiler component. In this way,

the length of the assignment descriptions can be minimized and students can receive immediate feedback on the correctness of their solution as they develop it. Many students are able to turn in completely correct solutions. I do not believe that so many of the solutions would be entirely correct if the students were not provided with a testing infrastructure but were instead required to read lengthy descriptions of, e.g., the syntax of the source language they are compiling.

Self Checks I have introduced numerous opportunities for the students to check their comprehension during lecture. After I demonstrate a solution that uses a novel algorithm or approach I halt the lecture and present the students with a similar problem. This allows the students to see for themselves whether they truly understand what has just been presented and allows me to identify common student misunderstandings.

Small Steps I have deliberately broken down the assignments into small, self-contained components. In previous compiler courses at the University of Wisconsin-Madison there have generally been about seven assignments; my course requires eighteen. I believe that there are several benefits to this. First, the students get into the habit of expecting that an assignment will be due every week; it becomes pointless to procrastinate. Second, it is possible to relate the assignments more clearly to the topics that are currently being discussed in class. Third, the assignments are broken down into more manageable chunks. Since no single assignment is of huge size and complexity a student is less likely to become overwhelmed and be unable to develop any sort of solution. Fourth, none of the assignments is of overwhelming importance. This frees me to have clear and firm grading standards; if a student truly wishes to do better they have the opportunity to learn from their mistakes on one assignment and improve their performance on the next. If one assignment depends on a previous assignment a solution for the previous assignment is always provided so that every student has an opportunity to start afresh.

Transparency The testing infrastructure I use to grade the assignments is exactly the testing infrastructure the students are provided with for their own testing. By this means the students can be assured that they will not lose points due to trivial misunderstandings and are able to focus on the essential principles that they are supposed to learn from the assignment. The testing infrastructure is also valuable for instruction; in interacting with a student I can suggest experiments to clarify assignment concepts.

Communication The students use the revision control system Subversion to develop and hand in their assignments. One goal of this policy is to make sure that every student learns how to use a revision control system. Students may find Subversion challenging at first; but they will rapidly find that it increases their efficiency and reduces the time they have to spend on assignments. Another goal is to allow efficient sharing of code

between them and me. The students are encouraged to commit their changes to their assignment repository before coming to office hours or emailing me with questions. I can check out from the repository and view their code directly and, if we make changes to the code together, commit these changes to their repository.

I consider the grading and testing infrastructure for an assignment when I design it. I know from my own experiences as a teaching assistant that when the grading standards for an assignment are ill-defined students often receive vague or even misleading answers to their questions about the assignment. Moreover, if it is not considered ahead of time the grading may turn out to be very time consuming and onerous so that students do not receive a grade until weeks or sometimes months after an assignment is completed. In my course all the assignments are returned promptly, sometimes within hours of the submission deadline.

Student Response I am pleased to say that the way I have put my principles into practice has been recognized and praised by a number of my students. The following are a few quotations from my recent evaluations:

- Assignments are very clear and efficient; I learn a lot from them and none of it seems like busywork.
- I enjoy the in-class examples that I do myself. They break up the lecture very well.
- Assignments are very good for increasing understanding of the material.
- Love all the mini-assignments instead of just having huge programs thrown at us.
- I've been taught PDAs and regular expressions three times but this is the first time I've truly understood them.
- Assignments are clear and graded in a timely fashion. Big plus.
- Great pace. Love the examples.

Respect I treat all of my students with respect and courtesy regardless of how well they are able to manage the material. I am very patient with students for whom English is a foreign language and work with them if they need extra help in understanding the material in assignments or in lectures.