

In *Second Annual University of Pittsburgh Teaching Excellence Conference:
Technology in Teaching, Pittsburgh, Pennsylvania, March 29, 1996*

TECHNOLOGY USE IN COMPUTER PROGRAMMING COURSES

Marek J. Druzdzel

Department of Information Science and Intelligent Systems Program

Programming is a skill that is taught primarily by practical examples and exercises — a programming course that offers none of the two is unlikely to be effective. Because computer programming is embedded in technology, teaching programming can be significantly enhanced by technology, probably more so than in most traditional disciplines. Paper handout and overhead projector slides can be replaced in the classroom by a computer projection panel. In addition, computer networks can give the students individual attention in their programming efforts and provide a classroom-like experience outside the class meetings. This is especially important when the level of preparation of individual students enrolled in the class varies. This draft describes my approach to teaching programming that strongly relies on computer use in and outside of the classroom. It has greatly enhanced the effectiveness of my teaching and has been met with a favorable reaction on the part of the students.

Computer in the Classroom

While some basic classroom tools, such as a blackboard and chalk, seem to be indispensable, classroom meetings can be enhanced greatly by using a computer with a projection panel. Seeing is believing: showing what it takes to edit, to compile, and to correct program errors (to “debug” the program) helps the students to make their first steps in programming assignments. It sometimes takes a simple programming experiment to understand such abstract concepts as compilation error, run-time error, stack overflow, recursion, or time and space complexity. When the time comes to discussing a complete algorithm or a program, its source, developed and tested in advance, can be viewed on the screen rather than on paper handout. The latter be replaced by an electronic pointer: the students can access the code after class and test it on their own — there is no need for a paper version. In addition, a computer in the classroom can be used as a motivational tool that shows the application of the techniques learned. In the *Data Structures and Algorithms* course that I taught recently, the programming assignments involved writing C code for symbolic mathematics. This was related to one of my research projects that concentrates on the development of a strategic planning system. Showing the power of a symbolic mathematics manipulation package (Mathematica), demonstrating a prototype of my system in the classroom, and subsequent explanation of how the assignments fit into the project were a major factor in motivating the students.

Communication Outside Classroom

A sizable fraction of the student population in the Department of Information Science are part time students. Many of these students, while intelligent and capable of working hard, had been out of school for several years. Some are making a major career change and may have little background in information science. As a result, teachers of introductory courses (such as programming or data structures) encounter a wide variety in student background and preparation. This poses a major challenge to effective teaching. If the standards are set high (I strongly believe that the students learn the most when the course work is challenging), those of the students who lack appropriate background need to work harder, especially in the beginning of the semester. They also need more personal attention than that provided by classroom meetings (to accommodate the needs of part-time students, our classes meet only once a week for straight three hours). My approach to helping these students with their efforts relies on building a social network that involves

their classmates, me, and the teaching assistant (TA). The courses that I teach have electronic World Wide Web class pages that contain links to students' electronic accounts and provide an easy way of contacting them. I also encourage the students during the first class to write down names and phone numbers of at least three of their classmates (just in case the computer network is down ...). This social network provides a support system on which the students can rely in case of difficulties with the material and the assignments. One way by which I stimulate collaboration among the students is my grading system: the grades in my classes are not curved and individual performance will not become better by the mere fact of poor performance of others. Programming projects are performed in teams of two to three students formed during the second class meeting. The teams are heterogeneous in terms of prior knowledge of their members. These are required to come from two distinct lists that I compose after the first class meeting on basis of an introductory questionnaire that asks questions about student background in programming (e.g., programming languages known, number of lines of code in the largest program independently written, etc.). I believe that group work is especially important in computer programming courses. Most real world programming projects involve teams, so the students exercise skills that will be important in their future work. In addition, their team colleagues dramatically enhance their learning experience and are the first and most powerful resource outside of class. Most problems, including programming errors, are solved within the team and only the hardest problems make it to me or to the TA. Team work allows for larger and more challenging assignments while reducing the overall workload. The second major element of the social network is communication by means of electronic mail (EMail) with me and with the TA. While all of my classes are based on EMail communication, I believe that EMail is particularly valuable in programming classes. All assignments are submitted by EMail. This allows for a true midnight deadline, not limited to the days when the class meets. Electronic submission is easier for the part time students — they do not need to come in physically to submit their work. The time stamp of the EMail message provides an unbiased and respected by the students arbitration of the submission date. The assignments are tested by the TA on sample data sets that are made publicly available later. Test results along with the final score are send back to students electronically within a week. In addition to the assignments, a major topic of EMail interaction are request for assistance in debugging and requests for clarification. A really valuable feature of EMail is that it is integrated in the students' normal working environment (they can send and receive EMail while working on their programs) and is active 24 hours a day. Students who are in distress (and this is not unusual among programming novices) when working on a problem in the evening or in the weekend can often get assistance within hours.

Concluding Remarks

An important factor in success using my approach is a knowledgeable TA, who ideally is an expert programmer and uses his or her EMail regularly. In addition, electronic communication, and especially electronic submission and grading of assignments, require some organization on the part of the instructor. For example, for any programming language, there are several compilers available on various computer systems. It is important to set a standard that the student submissions have to conform to. This, in itself, is an opportunity to teach the importance of standards and risks related to departures from these: system-dependent programming tricks do not transfer easily between systems. In my case, I choose our departmental mainframe computer to be the standard and place the responsibility on the students to test their programs on that computer before submission.