"Any clown can have the facts, but having opinions is an art."
Charles McCabe, *San Francisco Chronicle*

The Open Channel is exactly what the name implies: a forum for the free exchange of technical ideas. Try to hold your contributions to one page maximum in the final magazine format (about 1000 words).

We'll accept anything (short of libel or obscenity) so long as it's submitted by a member of the Computer Society. If it's really bizarre we may require you to get another member to cosponsor your item.

Send everything to Jim Haynes, Applied Sciences, UC Santa Cruz, CA 95064.

---

Too many programs

Ken Magel
University of Missouri-Rolla

There are too many PhD programs in computer science in the United States. This statement may seem absurd at first reading. There are about 75 computer science PhD programs, a number which has been growing slowly over the last 15 years. Currently, these programs produce approximately 350 PhD holders each year, but there are employment opportunities for more than 3000. Many companies have unfilled positions for which they have been recruiting for two years or more. And computer science departments are marking time or even dying at some universities because they cannot find qualified staff.

An increasing percentage of the best new PhDs are going into industry or leaving universities for industrial positions. The reasons are obvious. Industry offers much better pay (often $5000 more per year than a university) and better job security (most universities are cutting down on the granting of tenure and a few are even firing tenured faculty). The most important reason, however, is that industry can offer more exciting opportunities for research than most universities, where the computing equipment is out of date or not sufficiently available for large state-of-the-art research projects. The ancillary staff (programmers, secretaries, etc.) needed for a large project is not available, and the National Science Foundation and Department of Defense agencies are not giving many large research grants as they did in the 1960's.

Some computer science departments are marking time, or even dying, because they cannot compete with industry for qualified staff.

Presently, university computer science seems to be following the traditional pattern of mathematics (and physics and chemistry), i.e., a large number of small PhD programs. The majority of mathematics programs have 10 to 15 total students or even fewer. Many, if not most, sub-areas of mathematics are not covered by the available faculty. Thus, graduate students are forced to take courses and do their research only in the areas the faculty is interested in. Frequently, they have very little choice of courses, since almost all the students must take a course for it to meet minimum enrollment requirements.

Such limitations on course variety are increasingly characteristic of computer science PhD programs.

A reduction in the number of PhD programs would alleviate these problems. If there were only 30 programs, the present 800 to 1000 students would provide 30 students for each program. These should be enough to support a much greater variety of graduate courses than most programs now offer. The programs then would be in a better position to compete for still more students.

The problem of recruiting quality faculty would be alleviated, too, since only 30 programs would be competing for the best research-oriented faculty. These 30 programs should be able to add 3 to 10 good faculty members each. If these new faculty were carefully chosen, the 30 programs would each be able to cover most of the major areas of computer science.

With fewer programs the problem of outdated and inadequate equipment and insufficient ancillary staff also could be solved. Perhaps each program could be supported by one of the major companies with a substantial need for computer science graduates (e.g., IBM, General Motors). A symbiotic relationship between a company and a major computer science program could be established. The company would provide equipment, staff, possible state-of-the-art research problems, and some students. The university would provide a different perspective on the problem, academic credentials for some of the company's present staff, and future employees for the company. And the company could get a tax write-off for donated equipment.

Of course, there are philosophical and logistic problems involved in reducing from 70 to 30 computer science PhD programs. Nevertheless, if we do not formally reduce the number of programs, substantially the same effects will occur informally. The marginal programs will get still weaker; they will have even more outdated equipment and even more trouble attracting or keeping the best staff. The strong programs will continue to get stronger; they will find the industry, government, or foundation support necessary to update their equipment. The differences between marginal and strong programs will increase to the point where a PhD from a marginal program will mean something very different from a PhD from a strong program. By deliberately reducing the number of programs, we can avoid the long and wasteful death struggle of the weaker ones.
Dangerous languages, or... a time bomb waiting to go off

Vincent Manis
University of British Columbia
Vancouver, British Columbia

With considerable interest, I noted that the papers on program testing appearing in the August issue of Computer were primarily concerned with testing programs written in languages such as Fortran, Cobol, and Jovial. One would have to agree that a tested program in such a language is more trustworthy than an untested or insufficiently tested program. One would also have to concede that careful consideration might lead to programming standards which prohibit some of the most appalling blemishes in these languages—the EQUIVALENCE, arithmetic IF, and ASSIGN statements of Fortran, the ALTER statements and unbridled use of PERFORM statements of Cobol, the scalars and arrays with the same names in Basic. Programmers often do not understand the purpose of these rules. Furthermore, the rules themselves are often quite arbitrary. One local government agency even banned the PL/I DO WHILE construct.

Program testing simply becomes much more difficult when the needed control structures must be cobbled together out of spare parts, when data structures must be mapped by the programmer onto machine-oriented representations, when private variables are required by language rules to be far more public than necessary, or when the programmer is not even warned of a simple clerical error, such as a misspelled variable name or an omitted comma in a DO statement. "DO 10 i = 1 10" is a perfectly valid Fortran assignment statement.

If we are going to claim that programs to monitor intensive care systems in hospitals, to design and operate nuclear power plants, or to guide space probes are feasible goals, we had better realize that the vast software inventory written in these dangerous languages is not, as is often claimed, a significant economic asset, but rather a time bomb waiting to go off. Though simply changing the language in use will not solve all or even most of our software problems, the present situation seems to me comparable to the attempt to use highly sophisticated methodology to design and build a bridge made from pipe cleaners and processed cheese.

January 1980