tional fractionation of computer science into "pure" and "applied" components is undesirable, then greater attention will have to be devoted to making it possible for a computer professional to receive a Ph.D. or equivalent degree on the basis of a thesis that is primarily applied in nature. By this is meant a thesis which contributes not to computer science per se but to a nontrivial application of it in some field external to computer science. For example, an acceptable Ph.D. thesis of this nature might deal with the application of computers to medical diagnosis; or to problems in air traffic control; or to simulation of neural nets, etc.

It is essential that a student electing to do his dissertation in an applied field should devote a substantial amount of time to familiarizing himself with it. Thus, if his thesis is concerned with, say, the medical diagnosis by computer, he should be prepared to spend a month or two in a hospital acquainting himself with various aspects of medical diagnostics. The Ph.D. dissertation of G. A. Gorry* of Project MAC, M.I.T., is an excellent example of a thesis of this type.

In summary, although one can find isolated examples of Ph.D. theses in what might be called "applied" computer science in various institutions, this panelist believes that a conscious effort should be made to encourage work of this type within computer science and electrical engineering departments and accord it the same respectability as research in pure computer science.

The educational system may not respond to external pressures because it sees the need as temporary or non-critical, or it just may not be interested in such problems. The educational system may say it hasn't the resources to provide the professionals or settle for a merely adequate solution. It may even delay solving the problem by an act of generalization or systematization that, at best, functions to postpone.

The present graduate programs of computer science arose from the second process. As a result there is an aspect of computer education that is not being provided in response to a situation of the first kind. I refer of course to the education of a class of people that I shall call software engineers whose training and goals are quite different from those of our current computer scientists.

These badly needed people are engineers and their domain of specialization is software—its design, production, and servicing. Actually this is too restricted a view: Their specialization is computerware, both hard and soft. Actually they deal with a spectrum of states of computer matter each stable only in certain environments.

It requires little insight or sleuthing to see that such engineers are in very short supply and the need for them is acute. Accepting this state of affairs, certain questions need to be put and then to be answered by the educators and the employers of such people.

1. Should training be conducted in the university or in technical or trade or junior college schools or indeed in all?
2. If in a university, in what department or discipline or school should it be conducted?
3. If in a department or interdisciplinary program, at what levels should degree programs be offered and in what order of priority?
4. Do these programs exist in the steady state as a separate discipline or as a minor or option or modification of existing ones?
5. Is the need going to persist and, if so, will it persist in its present form? Will the people be educated for the coming generations of systems problems?
6. Why do we speak of engineering and not science?

It is here suggested that the proper place to start is at the master's level. From that program one can move up to the doctoral and down to the bachelor's programs. The master's program should not be a faceless one turning out computer scientists whose grasp and reach could not attain the computer science doctoral degree. The engineering component of this education is

---

**Identifying and developing curricula in software engineering**

by ALAN J. PERLIS

Carnegie-Mellon University
Pittsburgh, Pennsylvania

One basis for developing an education program is the recognition of a continuing need for a certain class of professionals in our society. The need may be redressed because:

An influential or significant part of the society may have a need for professionals that is not being met by the educational system.

An influential or significant part within the education system may observe an unrecognized need of society and begin to prepare what will eventually be needed.

The goal should be concentration on known tools and their effective use and not on periods of intense innovation or discovery. The choice from competing designs is more important than the discovery of the existence of designs. The issue of stability is more critical than that of growth and change. The determination of task magnitude is as important as the discovery of method. The directing of teams is as critical as the spark of breakthrough. The professional accomplishment of mean tasks that are of peripheral importance to the society are distinct from the devotion only to the bizarre, rare and new.

At the moment there are few studies being made of the problems of training these people. Rather than reproduce a curriculum here, I should like to list questions that a trained engineer should be able to answer:

1. Given a software task, similar to familiar ones, and a set of computers, evaluate the machines and the task to make a meaningful choice of machine and representation of the software system that is optimal. How? What is optimal? How relevant are the issues of manufacturer support? Compatibility? Stability? Natural gradients for change, growth, and improvement?
2. Given a software task, what is a rational schedule for its completion given various person

nel situations? How do you get, train, or even recognize adequate programmers? How do you set up work loads?
3. If n people are programming a system, what do you do when the n + 1st arrives, etc.?
4. How do you test a system? What kind of a system do you organize to handle and respond to pressures on a system?
5. How do you market a system? What makes a system useful? How do you copy someone else’s system?
6. How and what do you learn from building a system? What goes into the inventory stockpile after spending time on a software task?
7. What are the tools of the trade? How are they catalogued? Related to diverse hardware?
8. How do you tie together disjointed systems into coupled ones solving enlarged tasks?

I look upon the professional education of software engineers as an amalgam of mathematics, management science, computer science and practical experience gained from contact with actual software systems and associated problems. Some of the master’s may continue onto the doctorate in computer science, but the program is not seen as being merely preparation for a doctorate in computer science.