Chapter III

Computer Science Reaches Out-- teaching programming to the masses, 1965-1985

(part 1)

To read:

Lykos 214
Patton 325
Piscopo 342
350 re tenure issues.

A. Minimal motive for teaching undergraduates as a source of income-- research money.

The point to be made here will be that graduate students were already being employed in government-funded research to the extent that was desirable for them. There is a lot of literature about DARPA, because it led to practical products, but there were a lot of other programs as well, with different government agencies competing to have their own research establishments. There was even project Themis, with its land-grant style. Faculty had their consulting jobs on the side. Part-time computer engineering and software development were the major sources of income, not teaching.

The National Science Foundation did not phase out "facilities-supported computing center" grants until 1971. By this time, of course, computers would have been well on the way to becoming instututional overhead, rather than just research tools. By the end of the seventies, the NSF got back into the game, buying VAX's for Computer Science departments. (keenan, oh 217, pp. 21-22)

These latter type of grants, which by the 1980's were worth several million dollars over several years, covered a machine, and maintainance, and support costs, to pay computer center staff. Naturally, these jobs would have been used as de facto scholarships for deserving students. The NSF made a couple of dozen such grants, as "starters," to get schools into computer science.
After the first grant, they were expected to get money from somewhere else.

(Hedges, oh 221, pp. 8-9, 16)

Eventually, the number of undergraduates created a faculty shortage, and this became a justification for supporting research to keep the faculty from going off to industry (Hedges, oh 221, p. 14)

The big DARPA departments spent millions of dollars annually on computers. The more modest Prarie departments spent hundreds of thousands. Salaries were in the single thousands. Tuition was in the hundreds. Laboratory fees were in the single dollars. In short, there was no sensible economic motive to promote undergraduate computer education per se. Computers were the determining aspect of budgets, and tuition was insignificant in terms of computer costs.

What a computer center did need was bright young people to program and operate the computer, and generate more grant money.

B. Accidental Recruitment and Diplomas for Hackers.

However, as soon as computer centers were on campus, and not sealed off by security guards, undergraduaduates began infiltrating them. Undergraduates were less pragmatically interested in computers than graduate students-- they simply felt that computers were neat. They turned up, talked their way in, and were soon put to work. Over time, the undergraduates became steadily younger, and eventually presented a fait accompli to computer science departments which had intended to recruit only graduate students. Eventually, these departments were forced to set up undergraduate programs.

As a high school student and as a freshman at Harvard, in the mid-fifties, Peter Patton was interested in ancient, medieval, and non-western science-- in other words, in the roads not taken by western science. Harvard's stock answer for people with odd antiquarian interests was apparently that they should pick themselves a language to specialize in, and then pursue their interests through the lens of that language. Patton was sent to a professor who shared his interests, Professor (Daniel?) Ingalls. Ingalls happened to profess Sanskrit. Patton was the first new Sanskrit student in years. However, in his sophomore year, he discovered computing by chance:

> Yes, one evening about midnight I was coming home from the Sanskrit Library... you go by this building which was then called the Harvard Computation Laboratory; it was like a fishbowl and inside there were three computers: Mark I, Univac I, and, Mark IV. And so I went in. I saw a priest in there... [a] Catholic priest, because he was wearing a [cas]sock, and the contrast of this struck me. Here is a person wearing a uniform of the magistrate of ancient Rome working on this then very modern technology. I stood there at midnight looking at this brightly lit building. I couldn't resist, my curiosity overwhelmed me. So I went to the door and knocked... and the priest came to the door and said, yes, what did you want... I told him the reason for my curiosity. And he said, 'What do you do?' He said, 'Well I am working on the Book of Job.' And I said, 'On the computer?' And he says, 'Yes. Do you know the Book of Job?' I said, 'Yes, of course'... I started quoting the book of Job to him in Hebrew with an English translation after each line and he says,

> ‘Come in, Come in!’ (Laughter) And so he spent the evening, well the morning I guess by then, showing me how he had taken the book of Job and he had coded it in the Hebrew letters like 11, 12, 13, 14 and 15, two decimal digits, and he had written a program and he was trying to classify the tri-literal Semitic stems in Hebrew, into Elamite, Chaldee, Hebrew, Arabic stems because at that time, this would have been ’55, the approach for the Book of Job which has probably more hapax legomena, once said words, than any book in the Bible, except perhaps the Song of Solomon. But certainly the high point is the Divine utterance in the book of Job, chapter 38-41; since these words didn’t occur elsewhere in the Bible and the ancient Hebrew language is defined by the five thousand words, in the Hebrew Bible, what we the Christians call the Old Testament, hapax legomena had to be from some other language. And clearly the Book of Job is the oldest book in the Bible. Hebrew tradition says it was written down by Moses they thought that he learned it in the Sinai Desert in the forty years he was there and he wrote it before he wrote the five books of Moses. So this priest was trying to find the roots for sources, the lexigraphical sources, in the Book of Job. (BAB OH 325, p.4-6)

The next day, Patton went back to the computer center, and talked to Ken Iverson. In due course, he switched his major to Engineering and Applied Physics. (BAB OH 325, p. 3-6)

Gerald Sussman’s experience at MIT, circa 1966, was similar. As a freshman, he insinuated himself into Marvin Minsky’s computer lab while Minsky was not there, and discovered that he could play with the machinery without being bawled out by the graduate students. Minsky probably knew all about Sussman by jungle telegraph, but of course Sussman would not have known that. At any rate, one fine day, as Sussman was working on a program, Minsky turned up at his elbow. To Sussman’s surprize, Minsky did not chuck him out, but on the contrary asked about the program Sussman was writing, critiqued it, and offered Sussman a job on the spot. (Crevier, p.88)

By the early 1970’s the recruitment of talented students had worked its way down into high school. As Guy Steele observed: "I was familiar with MIT’s facilities because the people there were sort of tollerant of of young kids hanging around the computer labs."(p. 17) Steele started as an undergraduate at Harvard in the fall of 1972, and as a programmer at MIT’s Artificial intelligence lab the preceding summer. MIT had a program for teaching high-school students on saturdays. Steele, a student at Boston Latin School, had been involved with that for three or four years, that is, from the late 1960’s. Of course, it can be taken as read that a nominal saturday program worked out to writing code all week, and trying it out on the computer on saturday. By the time he graduated from high school, Steele had reached the point of writing a LISP interpreter, doing what would eventually be considered college-senior-level work.

(Jonathan Erickson, 2005 Dr. Dobb's Journal Excellence in Programming Award, p. 16; Jack J. Wohr, A Conversation with Guy Steele, Jr., pp. 17-22, Dr. Dobb's Journal, April 2005)

Undergraduates had simply turned up and started messing around with the computer. They had, in effect, invoked the faculty's premises, that good work had to be taken seriously, together with its creator. Eventually, the faculty had to decide what to do about them.

Once these undergraduate programmers were in place, the emergent Computer Science faculty found itself with the same kinds of obligations towards them that it had towards the graduate students. Undergraduates began to be waivered into graduate courses on an ad-hoc basis, and eventually, undergraduate courses were instituted, one at a time. The result was that individual departments set up bachelors degree programs so that the undergraduates could get credit for what they were doing. The simplest approach, of course, was simply to give such students enough verbal encouragement that they would complete the requirements for some existing program, such as mathematics. However, this did not always work. One of the driving forces for undergraduate computer science programs was the existence of undergraduate student-workers in the computer center, noted by Marvin Stein, who were not making progress towards degrees, and who were in due course snapped up by the corporate laboratories. As Stein remarked:

> Another way we were losing students was the following; as director of the Computer Center, I made it my business to conduct a developmental program. We began with undergraduate students and we put them to work as computer operators. But a part of their duty was to have in-house education, and to take the courses that I have mentioned [an introductory computing sequence and a numerical analysis sequence]. These students then progressed from operators: they became consultants, and some ultimately system programmers. This progression went along with their educational development as well. Of course we had attrition. I might appoint twenty students at the beginning of every year, have ten of them survive to their BA's, and of those, maybe five would come to work as graduate research assistants in the Computer Center. We would suddenly discover that they would leave because they were finding it difficult to pursue their graduate degrees along the lines that interested them. Perhaps our biggest competitor was Bell Labs because they would hire the students and then let them go back to some university for a year to get a [masters] degree. (Stein, oh 90, p. 39, also see p. 37 re courses.)

Note that Stein does not make a sharp distinction between undergraduates and graduate students as kinds of students. It must have been assumed that anyone in the computer center had the potential for at least a masters degree. About the only basis upon which Stein could have been awarding stipends was superior

performance in regular mathematics courses. His two course sequences were not pre-requisites but co-requisites of a computing center job. It was extraordinary that fifty percent of such selected students should fail to get their bachelors' degrees.

At the undergraduate level, there was even less of an a-priori case for computer science programs than there had been at the graduate level. In practice, mathematics tended to blur with the mathematically rigorous disciplines, such as physics, engineering, and now, computer science. It was understood that using more and better mathematics allowed one to approach problems in other fields in a more abstract way, and therefore saved time and effort in the long run.

A reasonable program in any of these mathematically-based fields included so much mathematics that the mathematics requirement (or at least the reccomendation for promising students) was only incrementally different from the minimal requirements for a mathematics degree. The latter were, after all, keyed to the better sort of future secondary school teacher. The mathematics requirements for a graduate degree in such a field as physics might actually amount to an undergraduate degree in mathematics. At the same time, the coursework in all of these fields tended to be heavily "front-loaded," with the most important courses, and most of the specific requirements, in the freshman and sophomore years. The idea was to keep the student working away at mathematics at full speed until he had learned all he was going to learn. It was feared that if the pace was too slow, the student might backslide between courses and even between classes, forgetting what he had learned. The ideal was to have the student spend a good three or four hours on mathematics and related subjects which exercised his mathematics (eg. physics), every single day from high school onwards, until he reached his stopping point. An unusually promising student, of the type whom professors and corporations competed for, was likely to secure exemption from the freshman courses, or to be allowed to "test out" of a course in the sequence somewhere. The result was that such a student would reach the junior and senior years with little in the way of particular remaining requirements. The tail end of an undergraduate program in mathematics or the sciences was much more anticlimatic than in the liberal arts.

In the early 1960's Jim Gray, as an undergraduate student in mathematics at Berkeley, was free to do all kinds of things. Up until the middle of his junior year, he was working as a grader in the mathematics department. At that point, the department gave him a reseach assistantship. Obviously, he must have been running years ahead of the average student and of the requirements. (p.11) As Gray commented: "If you are a faculty member, and you see a bright undergraduate, this is a very good sign. You try to grab those people because they are full of energy and don't know they are supposed to have a life." (OH 353, p. 11) Gray accumulated graduate courses in mathematics. He took courses in numerical analysis, which is both mathematics in the strictest sense of the word, and relevent to computers. He also

took graduate courses in the electrical engineering department, where they were doing theoretical computer science. (10-12) In short, departmental restrictions were basically not operative for someone who zoomed through the elementary courses. There was no rule, and there could hardly have been, saying that an unusually talented mathematics undergraduate had to apply for a graduate degree in mathematics if he did not want to.

Departmental restrictions tended to operate on the kind of student who, in order to specialize in computers, would need a dispensation from the usual sophomore mathematics courses. Such a student would literally not take a single mathematics course in common with the promising mathematics student. He would leave mathematics at approximately the point where the promising student's advanced placement cut in. The type of student on whom such regulations operated was the type of student whom the emergent computer science faculty would not particularly want, someone who merely took courses, and did not contribute to the program of research.

It is therefore somewhat mysterious that the desirable undergraduates should have drawn a line in the sand about doing the few additional mathematics courses required to get degrees. It would have been much more logical for them to make a production about having to meet the numerous liberal arts requirements. A possible explanation is that the students in the computer center, emergent hackers, were reverting to the type of the old corporate computer engineers.

The hackers and corporate computer engineers held, in essence, that there is no such thing as a higher esoteric academic knowlege. There was a certain logic to this view. The more highly theoretical and abstract methods of reasoning tended to lend themselves to being reallized as general-purpose programs, the classic example being the language compiler. It was not necessary or desirable for human programmers to do things the machine could do. The existence of such a program as the compiler meant that human programmers would go and do something else, and consequently, the formal reasoning leading to the compiler would not be applicable to whatever they were doing, for the time being at any rate. The hackers felt that if they should need to know some advanced topic, they could learn it from a book. According to their lights, one went to school as a means of gaining practice until one's general proficiency was such that one could gain apprentice employment. Once a student was given a job in the computer center, and began doing real work, he was more or less unwilling to go back and take classes. In these terms, being sent back to take classes was an insult, because it implied that the student could not work independently.

Paradoxially, students of the Hacker type were willing and even eager to take courses leading to skills of the "get-your-foot-in-the-door" variety, of which typing and shorthand is the classic example. The purpose of such courses was to enable the student to make himself immediately useful in a real workplace, while getting a chance to learn the business. In engineering, mechanical drawing traditionally played a role analogousy to typing and shorthand. In computer programming, it was the common computer languages. When COBOL became available,

hackers wanted to learn it, simply on the grounds that it looked like being the new shorthand. On the same principle, a hacker was likely to be a good sport about taking courses of no intellectual prentensions but obvious usefulness, such as public speaking.

The professors, of course, had the opposite bias. They were commited to the idea of a university, having foregone corporate pay scales to pursue this ideal. They were inclined to insist on a particular style of programming which was, insofar as possible, an extension of mathematics. The conventional sophomore course in mathematics is heavy on variations upon the theme of calculus: multivariable caluculus, diferential equations, laplace transforms, and even possibly exotica such as calculus of variations and integral equations. (cite courant) The emergent computer science professors did not insist on this, but were eventually willing to substitute a course in what came to be called "finite mathematics," that is abstract algebra, set theory, etc. topics which were normally taught at a somewhat more advanced level in the mathematics department. What they were not prepared to give up was the principle that the student should learn all mathematical topics relevent to a subject outside of mathematics, and that he should then follow it up with a course in the target subject which was couched in terms of this mathematics. As applied to computer science, this meant a course in theory of computation.

This impasse was never really resolved. The identity of the hacker persisted. Joseph Piscopo was an example of a type of student who was not willing to subscribe to the professors' ideals. He wanted to be a businessman, and he was also interested in computers. He did not start with the professorial assumption that commerce was at least slightly vulgar. In that light, he took what he wanted from computer science, and discarded the rest.

Piscopo, founder of Pansophic Systems and developer of the Panvalet file management program, went through the computer science program at the University of Illinois, graduating in 1965, while Computer Science was still part of the mathematics department. For him, the formal content of the program was secondary to the chance to actually work with computers:

> The mathematics background was not terribly relevant other than the fact that the computer programming courses at Illinois were all in the Mathematics department. More relevant though was, I learned how to program a variety of different machines at the university as well as preparing me for being able to adapt to any kind of a computer later on (Piscopo, oH 342, p. 2)

Piscopo's younger brother graduated in 1969, by which time the Computer Science department was independent, but it was substantially the same curriculum.

When Piscopo graduated, he went to work at the Joliet army ammunition plant for a year, and then moved on to Montgomery Ward, working on things like inventory systems. By 1969, he was

considering going to business school. At this point, his uncle arranged to set him up in business instead. The uncle gathered together twenty-five friends and family, who contributed a total of $150,000. Piscopo was apparently a hereditary businessman. His first product, Panvalet, was a program of a type which IBM was reluctant to produce for commercial reasons. In short, the reasoning behind it was commercial reasoning, not mathematical or computer science reasoning. The company started up with Piscopo's brother and college roommate, but thereafter, it recruited ordinary business programmers rather than computer science graduates. Subsequent expansion was based around aquisition, rather than internal reseach and development. (p. 1-2, 6, 11, 13-16)

Joseph Piscopo was a businessman who happened to be interested in computers. He was not an academic, nor, in any real sense, a computer scientist. He was willing to play along with academic computer scientists to a limited extent, and for a limited time, because they had the computers. When this arrangement ceased to be useful from his own internal standpoint, a business standpoint, he dropped the connection, and, as far as one can determine, never looked back.

The long-term effect of putting the computer where passing undergraduates could look in the window was to encourage them to simply drop in and become involved. Involving undergraduates in the short term meant managing their relationship with the university in the long term. Eventually, this meant dealing with talented students who did not want to make the same kinds of life-choices which the professors had made. The effort to hang on to such students resulted in academic programs catering to people who were not necessarily overly talented with computers.