Chapter 2

Building an Academic Research Community-- Computer Centers and Graduate Students

draft of July 24, 2005

Universities got involved in computing on their own terms, not those of outside agencies. When computers arrived in the university, they were, for the most part, used in such a fashion as to support the distinctive values of the university. At first computer centers were set up to provide computational support for academic research. Then computer-related topics were incorporated in existing academic fields. When computer science departments were finally set up, it would be to maintain the self-identity of the university in the face of corporate competition. The subject matter of computer science departments would shift around, in a search for something which was not too corporate in orientation.

// expand this to an introductory section

When universities got their own computers, the computer centers were likely to be extradepartmental, organized by whatever persons happened to have relevant knowledge, experience, and interest. The computer was usually paid for by government funds, though not necessarily military funds. Organized training in computer usage grew up around the computer center, simply as a necessary outgrowth of the computer's own functioning. As the computer got more complicated over time, so did the training.

The extreme government financed "contract research" style of

MIT was not replicated at most universities. It was more typical to spend a lot less money, and to focus more on the university's internal needs, rather than seeking to become primarily a government contractor. By 1960, clear differences of style had developed, according to the priority placed on obtaining military funding.

The computer center at Stanford University, set up in 1952-53, was funded by the engineering school, under Frederick Terman, and the Applied Mathematics and Statistics Laboratory. Each organization committed $25,000 per year, more or less as an overhead on their research programs. Most of their research programs, of course, would have been grant-funded,and the computer center was therefore indirectly paid for by military funding. The computer center started off with a Card Programmable Calculator, and got an IBM 650 in 1957. Training was more or less ad hoc at this stage. The computer center was rather more concerned with educating professors than with educating students. John Herriot, the computer center director, observed: "Of course, as you might expect, the students took to the computers much faster than the faculty."[1] In other words, he did not have to do anything special to get in as many students as his equipment could support. They simply arrived automatically.[2]

In 1955 (?), very shortly after the MIDAC machine at the

---

1. BAB OH 21, John Herriot, p. 7.
2. ibid, pp.4-8.

University of Michigan started up, Prof. John Carr of the Mathematics department organized a course in elementary programming and numerical analysis. Because the MIDAC machine was classified, Carr had to take student programs to the machine and run them himself. A newly arrived, newly minted, Ph. D. named Bernard Galler sat in on Carr's course. Carr apparently did not want to teach beginning programming over the long term, because he immediately set about grooming Galler as a programming teacher and researcher. Galler took over the programming course the following term. The MIDAC console had to be declassified so that Galler could go and work on it. The next year, the university got an IBM 650, nonclassified, and, unlike the MIDAC, located on campus instead of out at Willow Run. When Galler began pushing the limits of the IBM 650's performance, Carr introduced Galler to General Motors, where they had an IBM 701.[1]

The development of a distinctive style of "land-grant computing" is illustrated by the experience of the University of Minnesota. Funding was treated firmly as a means rather than an end. The emphasis was on getting computers widely used, rather than on doing contract research for the military. The university tended to worry about ethical considerations, rather than becoming captivated by the academic equivalent of business success.

The beginnings of computing at the University of Minnesota were striking in the extent to which the university chose a

---

1. BAB OH 236, Bernard A. Galler, pp.4-6,7, 13.

politics of its own, rather than simply having one imposed on it by apparent financial necessity. Minnesota drew on an incredible range of financial sources _other_ than the military funding favored by an institution such as MIT. Minnesota was consciously egalitarian, trying to spread knowledge of computing among the largest number of people. The university was being very consciously true to its land-grant origins.

Marvin Stein arrived at Minnesota in 1955, the same year that Bernard Galler arrived at Michigan. Stein was already an experienced computer programmer, and Engineering Research Associates (Univac) had effectively recruited him because ERA was making a substantial gift of computer time to the university, and someone was needed to see that it was used properly. The university's part of the bargain was to give Stein a tenure-track appointment in the Mathematics department of its Institute of Technology, and he was given tenure after a year.[1]

Stein launched a yearlong graduate level introductory course in programming. The demand was such that he had to teach two sections of fifty students each. Stein ran the class in "semi-seminar" mode. As he described it:

> I would suggest some type of exercise that would illustrate the ideas that I was discussing. Then we would have one student who would volunteer to illustrate that, and actually carry out the exercise on the computer. That student would write a report. All the other students would receive a copy of that report and their assignment was to do a critique of it.[2]

---

1. BAB OH 90, Marvin Stein, pp. 17, 24.

2. ibid, p. 19.

At the same time, another faculty member, Bill Munro, taught another sequence in numerical analysis. Both courses involved regular laboratory work, apart from preparing problems for reports. There was some kind of screening requirement for the programming course, simply to keep the numbers of students within manageable limits. Stein could not recall the details by the time he was interviewed, but the requirement was apparently sufficiently porous that anyone out of the ordinary who had a decent reason for wanting to learn about computers could get in. That included undergraduates, and liberal arts graduate students.[1]

Stein also made it known that he was available to help anyone with a research problem which might be amenable to the use of computers. However, he stipulated that the customers had to learn to do their own programming, albeit with the help of Stein and his staff. The result was that once a user did one project, he would be disposed to come back for others. These computations mounted up to a point beyond what ERA's initial gift would cover. So, in the more advanced phases of the projects, the researchers would go out to Convair in San Diego, where Stein and the university had a connection.[2]

This was not a long-term solution, of course, so the university obtained a NSF grant for $100,000, and found other

---

1. ibid, pp.17, 31-32.
2. ibid, pp. 17-19, 21.

monies from various sources, to a total of $250,000. They looked around, to see what they could get. There was a machine at Los Alamos which they were offered, but it turned out to be a fundamentally unreliable prototype.There was an offer from the newly independent Control Data, but that was still a paper company with a paper machine (this must have been approximately 1957-58 or thereabouts). In the end, Univac agreed to sell the university an 1103 machine for exactly the sum of money they had raised, rather than what it would actually cost. About 1960, the university got another computer, a Control Data 1604, with half a million dollars from the National Science Foundation, and a quarter of a million from the state legislature. In 1963, the university got a Control Data 6600. The nominal price was three millions, but Control Data discounted it to a million and a half, and the National Science Foundation kicked in $900,000. The state regents borrowed the remainder.[1]

Stein took the computer center on a consciously anti-commercial path. He stated his policy:

> One of our restraints was that we didn't care to be in competition with various manufacturers with whom we had to do business in other ways, and who were operating service bureaus. We allowed outside use if there was clearly some justification for it: if we had some type of unique program, or if it was one of our students who had received his or her degree and had gone to work for an outside organization, and wanted to come back to do something on the equipment that he or she had written the program for and was familiar with... I remember turning down offers from Honeywell to buy thousands of hours, primarily because it seemed that we ought not to be in competition in that way. Also, our faculty and

---

1. ibid, pp. 21-22, 27, 29.

> students were making good use of the time. We were pretty heavily loaded. We didn't want to get into a position where we essentially said, "Too bad for you, but we need the money." Our system was blind as to whether the user was paying for the usage or not. It might have been naive of us, but that is the way we operated. Students had as good access as research projects that paid. And I was a little worried that these outside users who were putting down large sums of money would demand priorities that at that time I didn't want to concede to them. Maybe if we needed the money, or if I knew more about money in those days, we would have done it; but we didn't.[1]

Make some reasonable substitutions, eg. high school for university, auto repair for computing, etc., and this speech could have been made by one of Garrison Keilor's Lake Wobegon characters. The mere expensiveness of the machine was no reason to run it according to corporate principles.

The nature of a technology is not defined by the circumstances of its first invention, but rather by the circumstances of its gradual adoption and modification. Academic computing was run by people who wanted to be part of the university, and who implemented that desire in hardware, when and as they could.

The first generation of graduate students interested in computers were still affiliated with regular academic departments. They met all the regular requirements for their disciplines. For the time being, the introduction of computers did not lead to conflict. However, the graduate students' advisors, themselves often returnees from the wartime computing programs, arranged for the students to get computer training

---

1. Ibid, pp. 27-28.

somewhere or other. This did not have to be on-campus; it might equally well consist in being sent to a summer job at a government laboratory somewhere. This course of instruction, beyond programming per se, came to consist of teaching the students how to build compilers and kindred programs, as a point of departure for linguistics research, the same point of departure that their mentors had reached while doing applied research in the employ of the military-industrial complex. Thus, this first generation replicated the precarious balance of their elders.

When computers first arrived, their influence was limited by their scarcity. Sometimes, graduate students got only very limited access to computers, computer centers, and the people who worked in them. The students got enough exposure to tantalize them, and to yield important long-term consequences, but in the short run, this was not enough to derail their graduate programs, and force institutional changes. The students got their Ph.D.'s before they started to act drastically on their new ideas about computers.

Thomas Keenan was a physics graduate student at Purdue in 1954, when the university ordered a computer. One gathers that the machine's arrival was too late for him to use it in his own research, for which he used desk calculators (he completed his doctorate in 1955). However he attended the training sessions, and became knowledgeable about computers. After graduation, he got a job at the University of Rochester, Rochester was also getting a computer, and Keenan was put in charge of the emergent

computer center.[1]

It took somewhat longer for computers to filter down to people who were not in recognized quantitative disciplines. In 1964, Bruce Buchanan, who had been a mathematics major as an undergraduate, was writing his dissertation in Philosophy at Michigan State University, on the subject of scientific discovery. He was trying to treat scientific discovery in a logical way, rather than as something ineffable. This is known, in the social sciences, as "operationalizing" a concept, reducing it to a model which is at least verbally specific (though not necessarily specific enough to stand up to being programmed). Buchanan had written the first half of his dissertation, which would probably have been a literature survey. That summer, Buchanan applied for a job as a policy analyst at System Development Corporation. He did not get the job, but his application got passed around to potentially interested parties--notably Edward Feigenbaum at the Rand Corporation. Feigenbaum, of course, was gearing up to launch the "expert systems" school of artificial intelligence, in the wake of Newell, Simon, and Shaw. He was naturally interested in anyone who was trying to reduce scientific work to something precise enough to be programmed. At RAND, Buchanan not only "...learned a lot about computing",[2] but was exposed to the unpublished or quasi-published works of Newell, Simon, and Shaw. He met people

---

1. BAB OH 217,Keenan p. 3.

2. BAB OH 230, Bruce Buchanan, p. 4.

with more nearly kindred interests than he could find in his home department. At the end of the summer, he went back to Michigan State and wrote the second half of his dissertation, along lines influences by Newell, Simon, and, Shaw lines. Buchanan then applied to Feigenbaum for a letter of reference. Feigenbaum offered him a job instead, and Buchanan accepted it, shelving his plans to teach Philosophy.[1]

At Michigan State, Computer Science was emerging as an undergraduate program, and as a branch of electrical engineering (, ref harry hedges, oh 221.). One does not know how much computer access Buchanan had before the summer of 1964, but he would have had to fight his way through all kinds of bureaucratic barriers to establish contact with the computer people on the Michigan State campus who might potentially be interested in his work.

In both cases, the incipient computer scientists were obliged to delay doing anything substantial about their new interests. This meant that potential conflicts with their original disciplines did not come out into the open.

As computers became more abundant, they were used especially by mathematics and hard science students. This, however, did not trigger conflict. Mathematicians and hard scientists were operating in the real world, not in an ideal one. Applied mathematics already existed before the computer. The effect of a shift to computer-based applied mathematics was to diminish the

---

1. Ibid, pp. 4-5.

role of routine calculation with adding machines, which the wartime experience had shown could be done by clerks. The work that students were doing with computers involved comparatively ambitious projects, which had a comparatively high mathematics content.

Gene Golub was recruited by the University of Illinois computer center in 1953, when he had just finished his bachelors' degree,[1] and got his Ph.D. in 1959, in mathematics. In the meantime, the mathematics department seems to have impinged on him very little. Golub observed that mathematicians were not as enthusiastic about computing as other fields,

> "...But there was none of the hostility that you would find at Stanford towards computing. I think people just realized that the computer was there but they didn't, there was no anger in their attitude towards computing."[2]

He must have taken the usual courses, examinations, etc., but they were apparently so uneventful as not to be worthy of mention. However, the major business of the computer center at Illinois seems to have been numerical analysis. This worked out to taking the mathematical unfinished business of the sixteenth to nineteenth centuries, and recasting it in terms of twentieth century mathematical orthodoxy. As such, numerical analysis is essentially conservative, like teaching mathematics, and was unlikely to attract strong animosities once the issues were

---

1. BAB OH 105, Golub, p. 4-5.

2. Ibid, p. 19.

properly understood.

Similarly, the mathematics department at Stanford had, by about 1958, set up an alternative form of masters degree for applied mathematicians, in which the student would program a numerical analysis problem, and write a report on it, rather than doing a traditional master's thesis.[1]

The situation was even clearer in physics. Physicists' approach to mathematics is of course results-oriented. They are, perforce, applied mathematicians on the side, and the only real question was what kind of applied mathematics.

Joseph Traub was a good example of a physicist in the process of becoming a computer scientist. Traub's family was a family of German Jewish emigres, with a long tradition of producing professional men such as rabbis and doctors. They had gotten out at about the last possible moment. Traub's formerly upper-middle-class father happened to be a bank official, one of those professions which does not travel well. He could only find marginal employment in the United States. This kind of family is sometimes called "sunken middle class," battered by circumstances, and waiting for a son to grow up and get through school so that the family can resume its former status.[2] Traub went to Bronx High School of Science, where he played chess as an extracurricular activity. He was not interested in ham radio, but he was interested in mountain climbing. In fact, his interests

---

1. BAB OH 21, John Herriot, p. 13.

2. See Jackson and Marsden, 1962, ch 2, section: "the sunken middle class," pp. 67-70, for a discussion of this phenomena.

were substantially the normative ones of a European schoolboy in a French Lycee or German Gymnasium.[1]

Traub went to City College on a Regent's Scholarship, living at home. Presumably the scholarship money went for his share of housekeeping expense. He majored in physics and minored in mathematics, taking advanced calculus from Emil Post. Post did not lecture, but conducted the class as a collective oral examination. This set a standard that Traub's graduate school coursework would fail to match in his eyes. Traub started graduate school in physics at Columbia in early 1954, with a teaching assistantship.[2]

Within a year or two, some time in 1955, Traub got involved in IBM's on-campus Watson Scientific Computation Laboratories. A friend told him about it, and suggested he go over, and it was apparently possible to just go in and talk to someone in authority. The Watson Laboratory had a bureaucratic alter ego as Columbia University's Committee on Applied Mathematics, on which the physics department, inter alia, was represented. In 1957, IBM gave Traub a generous fellowship, of about $2000, with unlimited computer time. He afterwards estimated that his thesis required something like a thousand hours of computer time.[3]

Meanwhile, the physics department per se was not engaging Traub's energies. Physics does not have comprehensive

---

1. BAB OH 70. Traub, pp. 3-10.
2. Ibid, pp. 11-14.
3. Ibid, pp. 14-16, 17-18.

examinations in the same sense that liberal arts fields do. The system of examinations and courses is actually a qualifying examination system, designed to insure that students learn a little about all of the branches of the discipline, and, further, this is typically spread out over three years, instead of being concentrated in the first year. There is only minimal opportunity for specialization in the formal coursework and examinations. The system is seemingly contrived to compel a very bright student to spend a couple of years messing around in a laboratory, instead of completing comprehensive exams in the first year, and doing his dissertation research in the second.

Traub was distinctly underimpressed by the academic side of the physics department:

> ... my feeling is that I learned a smattering of math, a smattering of physics, a smattering of numerical methods in school. But the way I really learned something is I would get interested in it because of my research, and then I would just gobble it up, or I would create things. I think in some ways it may have been an advantage that my formal training, either because I wasn't interested in somebody else's agenda or because I thought the teacher was so bad, was such a smattering. That, in fact, if anything, may have been helpful. But I do not feel like I had a good education.[1]

At any rate, Traub describes professors who distributed copies of their lecture notes, and then lectured from them, and gave exams in the undergraduate fashion, and turned a blind eye to

---

1. Ibid, p. 17.

class-cutting.[1]

Traub did his messing around in the IBM Watson Laboratory. Somewhere in the process, he ceased to be a physicist, but the requirements for comprehensive examinations were apparently sufficiently low that this did not interfere with his passing them. When it came time to propose a dissertation topic, he wanted to do chess (this being immediately after Arthur Samuel). This was not allowed, of course, but he was given an equation from physics to solve by numerical methods instead. In 1959, he finished his Ph.D. (under the Committee on Applied Mathematics), and went to work at Bell Labs.[2]

Traub was allowed to work on computers within a physics department, but not for purposes of ceremony. Provided he met certain requirements in orthodox physics, which did not by any means occupy all his time, the department would give him a credential based on these, and allow him to work on less orthodox subjects.

M. Granger Morgan is an example of a physicist who went through an even more complicated divergence. His story is indicative of the sheer extent to which physics departments would accommodate people diverging from the norm-- provided of course that they belonged to the small fraction of the population mathematically talented enough to be physicists. In the course of his undergraduate work (at Harvard), and the early stages of

---

1. Ibid, pp. 16-17.

2. Ibid, pp. 24-25, 28.

graduate school (at Cornell), Morgan discovered that he had all kinds of complicated humanistic interests. He managed to visit Latin America on the pretext of working at astronomical observatories in Peru and Puerto Rico, and then went off to do Latin American history at Berkeley. However, this did not suit him either. His humanistic interests were too eclectic, and too focused around science and technology. As Morgan explains:

> In those days there weren't doctoral programs like the one here [at Carnegie-Mellon] in Engineering and Public Policy, I knew I had to have a Ph.D. in something. I looked around and figured I could get a Ph.D. in applied physics faster than I could get one in anything else.[1]

Morgan's former advisor from Cornell had gone to set up a new department at the University of California at San Diego. Morgan followed, becoming the first or second Ph.D student. In the nature of things, his physics skills would have been highly portable. He would have been able to pick up quickly where he had left off when he had left Cornell, and he would not have had to cope with the kinds of complex identity crises which are the norm in the liberal arts. Morgan's new thesis advisor was himself in the process of becoming a computer scientist himself, by small increments, to the point that he was running the campus computer center. At any rate, Morgan, in his spare-time reflections, became convinced of the potential of computer programming as a

---

1. BAB OH 224, M. Granger Morgan, p. 4.

means of social mobility.[1]

In the last year of his doctoral work (Ph. D. 1968, per cmu department website), he launched a practical experiment. He found a group of underprivileged teenagers, employed in a federally funded make-work scheme, and arranged to teach them to program, using the computer in his laboratory. It was a roaring success. With his advisor's encouragement, Morgan began scaling up the program, and putting it on an institutional basis. This of course involved doing the work of a school administrator. He then offered a course in "technology and public policy." This again, would have been an obviously useful thing to do, giving the physics department an interesting and probably popular course for the liberal arts undergraduates to take in order to meet the science requirement.[2]

In due course, Morgan went on to the National Science Foundation, and eventually, the Engineering and Public Policy program at Carnegie-Mellon.[3]

Like Traub, Morgan was able to do things which were well beyond the scope of any reasonable definition of physics. In Morgan's case, this meant public education and public policy, as informed by scientific considerations. The physics department had no special claim to these fields, any more than any other science department. The largeness of public education and public

---

1. Ibid, p. 3-4.
2. Ibid, p. 4-5.
3. Ibid, p. 6-9.

policy practically make it necessary to treat all science and technology as one field, rather than breaking them up into specialities. If there had been someone in authority insisting on the "purity" of physics, his position might very well have become untenable.

Mathematics and Physics departments were willing to find common ground with the emergent computer scientists. There was practically always something that the emergence computer scientist could do, which was interesting as a computing problem, and was also desirable to the mathematics or physics department. However, there were often other departments willing to make an even better offer.

Some graduate students got involved in computing through departments where there was an understanding of carte blanche. Sometimes a department, or even an entire discipline, found itself with an intellectual vacuum. The extent of degrees and programs was often driven by the need to maintain social parity with other people, in other fields, whose intellectual requirements might be quite different. The department might be forced by circumstances to expand beyond what its subject matter could support, and if that happened, the department would be open to almost any presentable outside subject matter which would fill the gap.

For example, circa 1950-60, there was an understanding that a good research masters in engineering constituted full academic qualification. Given the basic engineering value of elegant simplicity, the Ph.D. in engineering is inherently a bit

contrived. It implies that the dissertation author spends a year without generating any results finished enough to publish. That may sometimes be necessary, but it is a situation to be avoided if possible. The ongoing development of abstract mathematical methods (of which computer techniques are paradoxically an extreme case) meant that there was less and less necessary knowledge for a student to learn, and correspondingly less justification for lengthening the curriculum. There is no real tradition of monographic writing in science and engineering generally-- the tradition is that of the journal article, often very short, and the fruit of a month's or a couple of weeks' work. To make matters more urgent, undergraduate engineering students customarily worked in a faster and more focused way than other science students. Additionally, engineering's underlying basic science was mostly from the sixteenth to nineteenth centuries. Engineering made relatively little use of modern physics. The underlying physics of engineering can be expressed in a page or two. The rest of engineering mathematics and physics is mathematics commentary, which could be drastically shrunken with more abstract methods of representation. The result was that an engineering student who went on to graduate school would be anything up to a couple of years more prepared, vis a vis the material, than a corresponding physics student. An engineering department, once it decided to start offering the Ph.D., had an intellectual gap to fill up, and could therefore be very catholic indeed in what it allowed students to do for a Ph.D.

Ralph Griswold, eventual founder of the Computer Science

department at the University of Arizona, and inventor of a couple of programming languages, is an example of the type of student such a department could sponsor. Griswold's father was a civil servant (State Department). Griswold majored in physics as an undergraduate at Stanford, more or less by accident. He spent his ROTC obligated service in the navy, teaching, by rote, Nuclear Warfare, a subject he had no interest in. On the basis of this experience, he decided that he did not want to be a teacher. When he went back to Stanford for graduate work, he chose the electrical engineering department, because the "... EE department looked like it was a place that would give the opportunity to get an unusually broad education."[1] His interests went as far afield as metaphysics, apart from the more mundane areas such as artificial intelligence. Once he got his degree, in 1962, Griswold moved on to Bell Labs.[2]

At the major autonomous engineering schools, the situation was even more extreme. The massive flow of grant money broke down the departmental system, superseding it with a system of laboratories run by senior professors, who had their own direct lines to the funding agencies. Under these conditions, it is practically difficult to tell who was a computer scientist, and at what date. Terry Allen Winograd arrived at MIT in 1967 with an undergraduate degree in mathematics from a small liberal arts college, followed by a year of linguistics study in England on a

---

1. BAB OH 256, Ralph Griswold, p. 5.
2. Ibid, pp. 3-5, 8-10.

Fulbright Fellowship. At MIT, he worked under Marvin Minsky and Seymour Papert in the Artificial Intelligence Laboratory. His dissertation, involving the "blocks world" was an exercise in computer driven linguistics. He failed to engage with MIT's premier linguist, Noam Chomsky, who was by this time in a state of feud with Minsky. For reasons of administrative convenience, Winograd's Ph.D. was awarded in Mathematics, and he was then given a job in Electrical Engineering.[1] Winograd remarked:

> I honestly don't know [about how he got an appointment in electrical engineering]. It's true with the funding, too. This was the milk-and-honey days. You didn't have to think about that; Minsky took care of that. If you needed slots, he got them. If you needed money, he got it. You just did your work. Junior people didn't think about those issues. So, I have no idea. I mean, I said, "Well, I'm done with my thesis, I'd like to stay on." And he said, "Okay, we'll make you an instructor."[2]

This was of course the nineteenth-century German "professor royalty" system with a vengeance.

The intellectual openness of graduate engineering programs grew out of the fact that engineering was in a state of intellectual implosion. The need to find new things to do outweighed any traditional criteria of legitimate subject boundaries. In the last analysis, engineering departments were willing to welcome in people who had no identifiable association with engineering, but who were willing to do humanistic things with computers.

---

1. BAB OH 237, Terry Allen Winograd, pp. 3, 5, 15, 25, 38.

2. Ibid, p. 26.

At the opposite end of the spectrum from engineering was Education. For institutional reasons, education schools had expanded far beyond their theoretical basis in psychology. The situation was opposite from that in engineering-- children are too complicated for theories and formulae to be of any use in dealing with them. Getting an advanced degree in education was something of an exercise in "ticket punching," in which teachers got a pay increment for having a masters, and school administrators were expected to have doctorates. As James Koerner documented in The Miseducation of American Teachers, an Ed. D. might very well work out to sending out questionnaires to school districts to ask how they used school busses.[1] In this climate, a graduate student who wanted to do something-- anything-- really well simply did not orthodoxy.

In the early 1960's (1963?), Dale Lafrenz enrolled as a graduate student in the mathematics department at Minnesota, having previously gotten a bachelors degree in mathematics education and spent a couple of years teaching. He soon discovered that he was not a mathematician, but rather a mathematics educator. He transferred to the education school, and got a job as an instructor in the university's "laboratory" high school. [the account is slightly unclear, but confirm this. states that he got a math degree at Marquette in the summers]. In 1963, he and his colleagues started teaching the high school

---

1. James D. Koerner, The Miseducation of American Teachers, pp. 180-192.

students to program computers.[1]

At the time, the usual and customary method of programming was to write programs, submit them to the computer center, and get a printout back, eventually. This did not fit very well with children's attention spans, of course. Lafrenz and his colleagues heard about John Kemeny at Dartmouth, made contact, and arranged to use his interactive computer system running BASIC. Kemeny's computer was made by General Electric, and the GE foundation came through with a grant to pay for the telephone charges to connect up from Minnesota. Eventually, Pillsbury in Minneapolis bought a copy of Dartmouth's software, and the University high school was able to use it, thus saving long-distance telephone charges. On the new terms, the education school group was able to scale up their project, and turn it into an outreach program.[2] Lafrenz spent two years running the outreach program, and then in 1970, he moved over to Honeywell, which had decided to get into the computer outreach business on a commercial basis.[3]

In both engineering and education, the logic of subject matter and credentialism resulted in the issuance of an "intellectual poaching license." This license was not perfect, of course, but it was good enough for students who would sooner or later be recruited by a major government or corporate laboratory.

The fact that graduate students were becoming interested in

---

1. BAB OH 315, Dale Lafrenz, pp. 4-5.
2. Ibid, pp. 6-11.
3. Ibid, pp. 11-15.

computers did not imply the emergence of computer science departments. Apart from anything else, the sheer scarcity of computers delayed the process for a few years. Even then, the departments containing potential recruits were able to negotiate a compromise. Finally, there were always some departments whose internal imperatives led to eclecticism, instead of leading to the formulation and definition of an orthodoxy.

Because of this lack of conflict in existing departments, it was a long time before even full-blown academic interest in computers would result in separate computer science departments. It was possible to find common ground with existing disciplines, even when doing things which were well within the core of what would become computer science. And if that did not suffice, there were always the corporate and government laboratories, which offered substantially academic working conditions. Setting up a department, or even an interdepartmental program, was a comparatively complicated proposition, not to be undertaken to solve a problem which could be solved by a professor calling up a friend at a laboratory somewhere to find a job for an ill-placed graduate student.

John Holland, at the University of Michigan, probably got the first Ph.D. in computer science. This was at the level of the near-accidental. In the late 1940’s, Arthur Burks had begun teaching automata theory in his philosophy classes, and had attracted students who wanted to do theses along those lines. Many of these students were in no sense of the word philosophers. As Burks describes it:

> Well there was a man on the faculty named Gordon Peterson, who had been a fellow student of mine at DePauw, in physics. He went to the University of Illinois, and then to the University of Louisiana, took a Ph.D. in physics and went to Bell Labs. Then he came here in the speech department after the war, and we became reacquainted. He had students in the physics of speech, phonetics, and acoustics and so forth. Then I began to teach automata theory even in the late ’40s in my philosophy of science and mathematical logic courses, and so I had some students who were interested in writing theses, basically in what we now call computer science, though we didn’t have that name. John Holland was the first of these students, and they clearly didn’t fit in our departments. That is, Holland wasn’t about to study two years of courses, to learn history of philosophy and other philosophy courses, in order to write a thesis on computing. And Gordon’s students didn’t fit in his speech department, which was oriented toward speech and drama. So as a consequence, Gordon and I organized or started to organize the joint program in Computer and Communication Sciences, bringing in other people. In 1957, we got permission from the graduate school to give masters degrees and Ph.D. degrees, even though we didn’t have any budget other than our research project budgets.[1]

This case arose only because Burks had a double identity as a consequence of the war, that of philosopher and engineer. If Burks had gotten his joint appointment in Philosophy and Engineering at the University of Pennsylvania, he might not have needed to go to such lengths, since, as we have noted, engineering schools were fundamentally permissive about what capable graduate students could do.

People differently situated did not face such an early crisis. For example, there were mathematicians doing theory of computation. Stephen Cook got his Ph.D. at Harvard in 1966, with

---

1. BAB OH 75, Arthur W. Burks, pp. 106-107.

a dissertation on the complexity of multiplication. There was a whole cluster of people around him doing similar work. Cook then got a position at UC Berkeley. He was denied tenure in 1970. By this time, his "natural colleagues tended to be in computer science departments."[1] He immediately got hired to tenure at Toronto, in Computer Science. By this time, of course, the convention was establishing itself that computer science departments were where computational theorists lived, and there must have been a sense that they should go and take those jobs, and leave the other jobs to other kinds of people.[2]

One of the givens was that there was an ongoing brain drain from the universities to the better government and corporate laboratories in all of the technically-oriented fields. The overwhelming majority of incipient computer scientists went off to a laboratory as soon as they got their Ph.D.'s. Likewise, the laboratories were not very particular about paper credentials. There were still only a handful of academic computer scientists or near-computer scientists-- perhaps several hundred or fewer advanced/committed graduate students in the middle sixties. By comparison, the corporations were huge, and could easily absorb anyone they could persuade to join them. A capable student in a relevant field who got into difficulties with academic rules and regulations could always go off to a laboratory. At Harvard, the mathematician and complexity theorist Alan Cobham actually

---

1. BAB OH, 350, Stephen Cook, p. 12.

2. Ibid, pp. 6-8, 12-13.

completed all the work for Ph.D., except for a secondary thesis in a minor field, but at that point, he went off to IBM.[1] At Minnesota, Marvin Stein's motive for eventually starting a computer science department was mostly to contain the brain drain, which resulted when students simply would not take course they did not see any use for, or prepare for examinations which likewise seemed pointless. As Stein put it:

> "I really felt that it would be difficult, if not impossible, to build up the staff properly and to retain the staff without a program, a solid program, in computer science in the university while the students that we had developed were going off to Bell Labs and other places."[2]

The result was that, by the early 1960's, after twenty-odd years of academic involvement with computers, there were still no computer science departments, and only a handful of interdepartmental degree programs. There were simply too many forces siphoning off potential students. Every institution wanted to get involved in computers, and every institution made what accommodations it could make without compromising its basic mission.

The emergence of computer science as an academic discipline was driven by the need of the new field to differentiate itself from its principle competitors for manpower, the industrial laboratories. At first, computer science was merely a response to

---

1. BAB OH 350, Stephen Cook, p. 6.
2. BAB OH 90, Marvin Stein, p. 40.

the technological obsolescence of applied mathematics as an academic field. This was reflected in the graduate-only character of the first programs. However, the need to retain graduate students led to an abortive foray into artificial intelligence. Attempts to approach problems in a theoretical way turned out to lead to alarmingly practical tools, which lead their developers back towards industry. By a process of trial and error, computer science drifted into the production of open source software, and the revival of engineering radicalism.

The break from the previous status quo came, oddly enough, from the most traditional form of academic computing, the use of computers in applied mathematics. New software was superseding the traditional role of the academic computer expert as applied mathematician. The applied mathematicians responded by going into computational linguistics. This was a stabilizing response, which avoided greater changes.

By the early 1960’s, the early and improvised computers were being replaced by standard computers of greatly improved performance produced by major corporations, and these computers case with service contracts and standard software, especially high-level programming languages such as FORTRAN and ALGOL. Such languages could be taught in a course of only two or three semester hours. An extensive course in applied mathematics could be covered in fifteen semester-hours, provided that the students could do their programming exercises in FORTRAN or ALGOL. The conditions were ripe, in short, for applied mathematics and its associated computer programming to be re-absorbed by

mathematics. On the whole, FORTRAN or ALGOL code compared reasonably favorably in "mathematical density" with other forms of mathematical writing. Problems which contained little in the way of mathematical novelty could now be dealt with in an appropriate place-- undergraduate courses.This would become even more emphatically the case in a few years when new languages such as SAS and SIMSCRIPT came to embody the specific technique of large sections of applied mathematics. Like the engineer in Kurt Vonnegut's Player Piano, the computer experts had invented themselves out of a job. Thanks to their efforts, there was no longer a mathematical justification for a separate field of computers.

Computer science, perforce, defined itself in terms which were, at first sight, only tangentially related to the things which computers were actually being used for in the university. That is, computer science was defined around computational linguistics, and eventually, artificial intelligence. However, what computers were actually being used for was applied mathematics. As Gupta[1] notes, following Conte, eight out of ten of the members of the computer science department at Purdue, circa 1964, held joint appointments in the mathematics department. Only a handful of disciplines used mathematics extensively enough to get in the position of formulating mathematical problems which were not readily solvable by paper and pencil methods. These disciplines tended to train their own

---

1. G. K. Gupt, "Computer Science Curriculum..., 2004, p. 6

applied mathematicians, many of whom would eventually become computer scientists. While George Forsythe claimed a diverse background for the members of the Computer Science department, nine out of ten had Ph.D's in either mathematics, physics, or engineering.[1] They came from the disciplines which cared about mathematics enough to really make their members learn mathematics.

The first aspect of computer science, computational linguistics, was about creating programming languages. This was at least a reasonable outgrowth of applied mathematics in the sense that a programming language could be a sort of framework into which to install applied mathematics software. It would give the emerging computer scientists a breathing space to adapt to new conditions. The identity of the applied mathematicians in the process of becoming computer scientists was probably as much a matter of social relations of production as it was of subject matter per se. They had all the organizational baggage of computers, and funding, and staff to use the computers. It was not very important if the computer center started producing software instead of calculations. If an applied mathematician could learn computer science by small increments over a period of years as it was invented, while continuing to work with the same people, maintaining the same standards of mutuality and cooperation, he might become a computer scientist without ever experiencing any sense of change. Computer Science was rather a

---

1. Ibid.

case of "things must change to stay the same."

From the standpoint of running the new graduate programs in Computer Science, there was no immediate need for undergraduate programs. Computer Science graduate students were naturally selected from persons already possessing sufficient liberal education and mathematical and linguistic background. The necessary requirements in those areas for an academic researcher were already compatible with undergraduate curricula. That is, a prospective Computer Science student could earn a bachelor's degree in one of the antecedent fields, such as mathematics or psychology, without thereby overspecializing, and could then set about learning other fields relating to Computer Science. Thus, there was no real need to set up undergraduate Computer Science programs. Some computer science departments (notably Stanford) acted on that principle, and stuck to it. As William F. Miller of Stanford explained:

> We spent a lot of time making sure our graduate students were well prepared. We did not develop an undergraduate major; we had curses [sic] for undergraduates, but we concentrated on graduates and research areas as far as our majors were concerned. I think that was a good move on our part... So, I wouldn't find it necessary to find a computer science major. I think our program here of a major in the mathematical sciences where students can take a concentration in mathematics or statistics or operations research or computer science but still get a broader major is adequate preparation for any of those subjects. I'd probably stick with that idea.[1]

As late as 1967, this view retained popularity, and the reports

---

1. BAB OH 29, William F. Miller, pp. 7-8.

on undergraduate curricula were challenged on the same grounds. (note for example L. Fulkerson's letter, CACM, mar 1967).

This was in keeping with the broader strain of academic thinking. The authors of academic white papers tended to view the undergraduate professional program, on the model of engineering, as an anachronism. The whole sense was "become a liberal art or get out." In 1968, Lewis B. Mayhew, who was professor of higher education at Stanford, wrote, in his "The Future Undergraduate Curriculum,"(1968) that by 1980:

> vocational training will gradually cease being a major preoccupation of the undergraduate schools. Much of the technical training needed even in such complex fields as electronics will be provided by employers who alone will be able to provide the newest equipment with which to conduct training.[1]

The minor liberal arts professions, such as education and journalism were to be moved up into graduate school, on the model of the MBA in business. In any case, Mayhew emphasized that "possibly five or ten percent of the adult population can man the entire productive enterprise."[2] Under the circumstances, undergraduates did not look like a particularly impressive labor pool.

The immediate requirement facing Computer Science was simply to find a way to hang onto the founders' graduate students, who at this stage, were mostly doing applied mathematics. The

---

1. Mayhew, in Eurich, 1968, p. 210.
2. Ibid, p 211.

immediate adaptation was to have them do much the same kind of work that they had been doing, only in a more ambitious form to reflect the better programming languages which were now available. Undergraduates were left out for the time being, because they did not present a key labor shortage issue.

Part of the ongoing identity crisis of academic computer science was how to differentiate itself from industry. Graduate students were perpetually being recruited by industry, with extremely good pay and working conditions. The very nature of an academic department mode it impossible to match the terms offered by industry. The founding of computer science departments did not solve this problem. At most, students waited until finishing their Ph.D.’s before going off to industry. To hang on to them, to maintain a collective group identity, it was necessary to come up with something that corporations could not or would not do. Artificial intelligence, the more radical fork of computer science, was, among other things, an attempt to meet this requirement. Computer science had to be defined as a liberal art, rather than an engineering discipline, because the liberal arts aspects were the only real reason for working in a poorly funded university rather than a munificent government or corporate laboratory.

The early artificial intelligence researchers were not in universities. Artificial intelligence was nothing if not prodigal of computer power, and the artificial intelligence researchers tended to go someplace where they could have more direct access to powerful computers. When the applied

mathematicians in the universities decided that artificial intelligence was going to be the capstone of computer science, they had to import artificial intelligence researchers, as in the case of George Forsythe bringing in Edward Feigenbaum at Stanford in September 1964, fully three months before the Computer Science department came into being. Similarly, John McCarthy had been recruited from MIT to Stanford in 1962, with the offer of a laboratory of his own. The artificial intelligence experts must have been more than usually difficult to recruit--no doubt it was necessary to have everything lined up, with a department in being, and the funding in place, before they would commit themselves.[1]

The championing of a computer science rooted in artificial intelligence was not the sort of organic growth that programming languages were. However, it was seemingly well suited to maintaining the identity of Computer Science departments against corporations. Artificial intelligence tended to draw on the full range of the university, not just on the technical faculties. Then too, businesses were more inclined to be leery about extravagant claims. A business's culture was in some part formed by people who handled other people's money, and who were inclined to worry about not crossing a line representing embezzlement. The computer science department could simply step up the ambitiousness of projects until the results became so erratic that the corporations could not follow. It would have appeared

---

1. BAB OH 21, John Herriot, p. 9; Crevier, pp. 64-65.

that costs could be kept within bounds by insisting that ideas be talked to death before actually being tried out.

A series of "manifesti" were issued, the first being Louis Fein's 1957 report for Stanford. There was a steady flow of similar works over the next ten years, mostly published in the Communications of the ACM.[1]

The details of these reports did not differ very importantly. The gist was that computer science claimed to be a science of information. But of course, essentially everything except energy is information. Thus, Computer Science was claiming authority over the thought processes of researchers in all other fields, setting itself up as a kind of "glass bead game," along the lines satirized by Hermann Hesse in Magister Ludi. Such a game implied turning all the humanities into Big Science.

John Holland at Michigan actually seems to have been aware of the Glass Bead Game analogy,[2] and to have consciously taken it as an ideal. Holland apparently did not take any notice of Hesse's satirical intent, or his ominous warning that knowledge could not be lifted from its context. Hesse's character of "Fritz Tegularius," the superlatively skilled glass bead game player with a deeply flawed character, is generally taken to be Friedrich Nietzsche. Hesse lays great emphasis on the idea that the delocalized intellect is not just erroneous, but morally untrustworthy. The singular thing about Holland's adoption of

---

1. Gupta, 2004, above, p. 2. Gupta provides an extensive compendium of the manifesti.

2. Waldrop, Complexity, 1992, p. 163.

Magister Ludi was that he did not engage this point. Theodore Ziolkowski, in his introduction to the Winston translation of Magister Ludi, refers to the "... humorless readers who complained to Hesse that they had invented the Game before he put it into his novel-- Hesse actually received letters asserting this!"[1] Most probably, Holland did not take the glass bead game very seriously. His actual feelings about big science can be gathered from the fact that in 1977, he spent $3000 for his own computer, on the grounds that this gave him better computer access than trying to use the university's machines.[2] This could only be true if he had completely refused to play the grantsmanship game, even at the departmental level. At a very rough estimate, his computer might have been a hundred times slower than a representative minicomputer of the same date. Holland would probably have had to let his computer run overnight to give him the answer which a minicomputer would give in five minutes or so.

The pretensions of computer science, or of mathematics for that matter, did not make a very great lodgement in the social science and humanities. There was a tendency for second-rate work to use statistics as "padding," but the hard-edged marxist moment had been back in the 1930's. By the 1960's, there was little backing for the kind of mechanical schemata which lent themselves to programming. This was not important, however, because the real

---

1. Hesse, Magister Ludi, p. ix.

2. Waldrop, above, p. 190.

question was the relationship of academic computer programming to industry.

In the aftermath of Noam Chomsky's Syntactic Structures, there had been a number of experiments with natural language processing, notable a group of dissertations supervised by Marvin Minsky between 1963 and 1966, which were afterwards published as Semantic Information Processing. This work looked more impressive at the time than it does in hindsight. The experimenters could plead obvious machine limitations. The programs could not necessarily handle any input of the type they were supposed to, but only those inputs which the experimenter had done some tinkering with.[1]

Another antecedent was Newell, Simon, and Shaw's Logic Theorist, mentioned above. Many pure mathematicians tended to discount the importance of Logic Theorist, placing less stress on the actual proving of theorems, and more on the decision of what to attempt to prove.

The sum and total of all of these was ideally to make computer science indigestible by industry. A graduate student who went into industry, it was hoped, would have to leave the esoteric knowledge behind. This solution had the inherent problem that the esoteric knowledge was not very practical.

The hopes of artificial intelligence did not, in practice, materialize. It did not become the kind of energetic philosophical debating society that proponents such as Forsythe

---

1. Crevier, p. 78.

had hoped for. What actually happened was that artificial intelligence became a near monopoly of a handful of schools and laboratories, which were willing to make more or less unlimited concessions in order to develop artificial intelligence programs.

Experimental work in artificial intelligence (as distinct from theoretical work) required virtually unlimited computational horsepower. Leaving aside the discouraging estimates of critics such as Stanley Jaki, even seemingly modest and reasonable undertakings presented serious hardware difficulties. A typical book occupies about a megabyte. With the core memory used circa 1964-68, a megabyte worked out to someone having to thread a wire through a metal "eye" nearly thirty million times, something approximating a lifetime’s work. Contemporary handbooks refer to memory by the byte, that is 1024, 2048, 4096, or even 8192 bytes. Given these economics, using computer storage in lieu of paper was unthinkable. Under these conditions, the most obviously reasonable measures, such as using a standard vade meccum or reference work (such as a pocket dictionary) in its entirety, were extravagantly costly. Other components were proportionately expensive.[1]

Such experimental work in artificial intelligence got "channelized" into a handful of laboratories on a handful of campuses which were prepared to go to extravagant lengths to get the requisite large sums of money. Other schools could not even make it into the starting gate, as far as artificial intelligence

---

1. See Crevier, p. 310, for comparison with the 1980’s.

was concerned. Artificial intelligence research probably did not have any more strings than any other kind of expensive research, such as atom smashers (see Nual Pharr Davies, Lawrence and Oppenheimer). However, someone who did not enjoy working for a defense contractor probably would not enjoy working in a well-funded artificial intelligence laboratory either.

It was hardly surprising that generous military funding would come with some strings attached. Paul Edwards has argued that artificial intelligence was directed toward the symbolic goals of militarism, as an expensive piece of political theater. Artificial Intelligence was to project an aura of machinelike infallibility, papering over the real risks of war. This, by itself, did not have to be crippling-- as indicated by computer science's later engagement with the even more theatrical video game industry-- but it raised the "defense contractor" question in a different form: why not go to work for Universal Studios and make a lot of money?[1]

Equally to the point, military artificial intelligence projects were conspicuously lacking in "engineering practicality." For example, vast sums were spent on voice recognition when the larger problem was usually best solved by inexpensive feedback automation. For example, it is a supremely challenging problem to get a machine to understand that an airplane pilot has just said "fifteen degrees flaps," and meant it as a command. It is much simpler to intercouple the flaps

---

1. Edwards, pp 142-43, 264-66, 299-301.

control to the airspeed indicator, etc., giving an effect similar to an automatic transmission in an automobile. The problem with this is that it tends to destroy the illusion that the pilot is flying the airplane. Most of the necessary human inputs to complex control systems ultimately boil down to either pushing a button, giving the machine permission to carry on, or not pushing the button.

To be successful in the sense of retaining funding, an artificial intelligence researcher had to do a good deal of undignified scrambling, making unkeepable promises, playing to the vanity of officials involved in funding. By the standards of physics in its 1930's and 1940's heyday, none of this was especially blatant. By the 1960's, the choice of funding sources was broader, and researchers had room to make choices. However, the basic proposition remained: if one is doing research which requires immensely expensive equipment, one must needs be a financier. Given that the results of research are by definition unpredictable, it is further necessary in this instance to be a disreputable financier. For an academic whose basic problem was to distance himself from business, the realities of fund-raising were a "show-stopper."[1]

Artificial intelligence was confined to a handful of schools, partly on account of its cost, and partly on account of the strains it placed on researchers grounds for being in academia in the first place. Basic disputes often express themselves in

---

1. For a discussion of puffery in artificial intelligence, see Crevier, pp. 6-7, 83.

seeming trivialities. At Stanford, which was divided between people who were comfortable with big science and people who were not comfortable with bit science, the conflict erupted in a dispute over a picnic. Pamela McCorduck, who had been Edward Feigenbaum's secretary, remembered while interviewing Alexandra Forsythe:

> In fact, I remember just after Ed Feigenbaum became the computer center director, I think he may have been acting director then, it was decided that there would be a picnic and it would be for the computer center, but not for the computer science or visa-versa, I don't remember. And I remember George [Forsythe] coming in to see me, because I was Feigenbaum's secretary then, and saying that his was very embarrassing. "How can we invite one side and not the other." And, wherever this was going to be held could only accommodate the one group and I said, "Well, I don't know, but that's the way it has been arranged" and he said, "Well, which one are you coming as?"[1]

Alexandra Forsythe replied that: "He was very much against discrimination or shutting anybody out of anything. He always wanted to include anybody and that was a very good trait."[2] It was on this sort of grounds that Artificial Intelligence became morally impossible for a large strata of computer scientists to pursue.

Schools which did not choose to go the necessary length to get huge sums of DARPA funding for artificial intelligence found themselves things to do which did not cost very much money. There was theoretical work, of course. However, the dominant tendency

---

1. BAB OH 17, Alexandra Forsythe, p. 18.

2. Ibid.

was to solve fairly practical problems involving software tools, for use with the type of computer which was readily available and not too expensive. These "Prairie Schools" (as distinct from DARPA schools), mostly Midwestern land grant universities, first started working on accessories for the computer itself, typically general-purpose software such as programming languages. When that got too similar to what the corporations were doing, they branched out into applications in nonquantitative fields. However, finally, they found their way to a political populist approach in keeping with the schools' ancestral traditions.

Probably the most common area of work was basic research applying to the major hardware and software used in computing. Basic research was supposed to be sufficiently impractical that corporations would not do it. However, things did not work quite that way in computing.

At Michigan, Bernard Galler's Ph.D. students did things like:

> ...a very ingenious linear programming technique... a thesis on disk cacheing and so on... a nice study of imaging techniques for medical images and scanner images... a nice thesis on dynamic updating of computer modules... the analysis of musical sound... an interesting topic involved with name directory service and organization of database systems connected with the X.500 standard... Intelligent Vehicle Highway Systems.[1]

These were nearly all things which could equally well have been done at IBM or AT&T. Many of these students went on to places

---

1. BAB OH 236, Bernard A. Galler, pp. 10-12.

like Texas Instruments, Apollo Computer, and Bellcore (AT&T).[1]

This kind of work tended to collapse the distinction between academic computer scientists and industry. There were universities which vended commercial products.

Jim Gray was one of the first computer science graduate students at the University of California at Berkeley. He first enrolled as a freshman in 1961, co-oping his way through college, and at one point, simply going and working in industry for six months. On his return, he got interested in computing. He not only took the authorized courses, but talked his way into graduate-level electrical engineering courses. The Mathematics department found him as well, and gave him research assistantships as an undergraduate. By the end of his senior year, he had exceeded the requirements for a bachelors' degree, and was a fair way along towards a masters. At this point, he went off to Bell Labs. Bell labs sent him to the Courant Institute at New York University, two days a week. After a year, Gray had accumulated enough savings to drop out and travel for a bit-- this being the age of Easy Rider, after all. Travel did not turn out to be as much fun as Gray thought it would be. So he went back to Berkeley as a graduate student under his old undergraduate advisor, Mike Harrison, who had moved from the Mathematics department to the new Computer Science department. Gray became interested in the theoretical aspects of programming languages and compilers, and wrote his dissertation on one aspect

---

1. Ibid.

of the subject. This sort of work was theoretical, but it was also extremely practical. The ultimate implication was the possibility of a "universal compiler," which could load a grammar for a language, and a program written in that language, and run them. IBM had every reason to be interested in someone doing this kind of work. Gray had also gotten interested in Jay Forrester's Limits To Growth, and it turned out that IBM Research was being expected to respond constructively to Forrester. IBM gave Gray a two-year post-doctoral fellowship at Berkeley, and then hired him at their Yorktown Heights research center. Gray found eastern winters depressing, and the only way IBM could hang on to him was to give him a job at its San Jose center, back in California, where he started working on research into relational databases.[1]

Computer science was facing, relatively early, a problem which would ultimately confront all of the hard sciences. If one can formulate a theory, however abstract, in explicit terms, then a computer can mechanically crunch out particular derivations and special cases of the theory, and apply then to particular problems. If the theory cannot be formulated in explicit terms, then it is not possible to determine whether a piece of evidence contradicts the theory-- in other words, the theory is not falsifiable, and is therefore not a theory, just a theological dogma. One implication of this phenomena was that there was no clear distinction between successful pure research on the one

---

1. BAB OH 353, Jim Gray, pp. 5, 8. 10-20, 23-33.

hand, and technology on the other hand. A pure science faculty cannot, on the basis of subject matter, avoid the kind of "revolving door" employment pattern characteristic of an engineering school, with people constantly going back and forth to and from industry. Pure research did not create social separation from industry.

Another possible mode of escape was to work on creating applied tools for previously unrepresented fields, such as art. However, the role of academic computer scientists was outweighed by that of corporate researchers. Success in such work put its creator rather in the position of a successful typewriter inventor, and encouraged him to go into the business of manufacturing his invention.

Charles A. Csuri came back to Ohio State University from the Second World War, got his bachelors degree in art, and his MFA, in short order, and joined the faculty. By the time he got interested in computers, he was long since tenured. In 1955, he first began to learn about computers through a personal friendship:

> At the University, a personal friend, Jack Mitten, who was a professor of engineering, began explaining to me computers and their applications for science and engineering. I asked all of those initial questions, like, 'What is a computer?' and 'How does it work?'. He patiently explained to me basic ideas and we began a dialogue about computers and art which continued over a period of eight years. At first he would describe the problems of converting by computer programming the Russian language into the English language, and that was a radical idea in 1955 - it may still be a radical idea. Please keep in mind that in 1955 there were no plotters or graphics output devices, but I was able to speculate about computerized theories of art and notions about artificial intelligence. The ideas were

> interesting, but the practical reality of programming prevented me from taking any serious action. We were close personal and social friends and our families frequently ate dinner together. During the cocktail hour, we would talk about computers, and as the martinis began to flow more freely, so did our ideas about computers and art.[1]

When, in 1965, Csuri heard of people producing "typewriter graphics" images with a computer, he took that as a signal that the time was ripe to get actively involved in computers. He took a short programming course, and started working away. When he needed money for programmers and special equipment, he went to the National Science Foundation. By small increments, Csuri moved away from actual art into research in computer graphics. This research turned out to be toolmaking. Like most of the early computer workers, Csuri's first task was to make his own tools for his own use, and that turned out to take on a life of its own. Thus his research into the possibility of doing art by computer had applied dimensions, such as special effects for television and movie studios. Csuri wound up with, for a time, his own company on the side.[2]

Csuri's work represented the outer limits of what the National Science Foundation could fund. Frederick Weingarten at NSF justified the funding on the grounds that he "wanted to see what somebody coming in from the outside making different kind of demands on the graphical capabilities of computers would do."--[3]

---

1. BAB OH 180, Charles A. Csuri, p. 4.

2. BAB OH 180, Charles A. Csuri, pp 5, 14, 15-31.

3. BAB OH 212, Frederick Weingarten, p. 12.

in other words, as a lever to enforce the rapid development of computer graphics. Despite a precautionary warning to Csuri to keep quiet about the business, the matter got brought up in congressional appropriations hearings.[1]

However, art was not only for academics. At about the same time as Csuri, a group of people at Bell Labs were also doing computer art: Kenneth C. Knowlton, Leon D. Harmon, Lillian Schwartz, A. Michael Noll, M.R. Schroeder, and Bela Julesz. As Jasia Reichardt, compiler of a contemporary anthology of computer art, observed:

> If one were to look for any one centre [sic] which has produced more, and a greater variety of computer-generated images than any other, one would probably have to turn one's steps to New Jersey and the Bell Telephone Laboratories.[2]

IBM also had its computer artists, working at the IBM Scientific Data Center in Tokyo. Among other projects, they rigged up an array of photocells, plugged into an art-producing computer, and then they got in a couple of ballet dancers to perform in front of the photocells for the benefit of the computer. This won them points for panache.[3]

The artists who were willing to work with computers at an early stage tended to have a strong preoccupation with artistic

---

1. Weingarten, above, p. 12; Csuri, above, p. 20.

2. Reichardt, The Computer in Art, 1971, p. 20.

3. Reichardt, above, pp. 20-33 (for Bell Labs), 81-87 (for IBM).

technique, and their goal was to produce universal tools for applying technique-- Csuri, when interviewed in 1989, was making a case for a program more or less like Photoshop. (Csuri, above, pp. 8,24) The samples of computer art in Reichardt's book are remarkably devoid of human content. They remind one of the exercises in which budding artists are made to draw circles over and over again, in order to gain fine motor control. A BFA final show will be full of paintings of arrangements of circles and squares, etc., and this is very much the flavor of early computer art. However, the same mental tendency which ordinarily produced abstract paintings in this case produced software tools, which could be used for all kinds of artistic work, including commercial art.

The corporations could do about as good a job of sponsoring computer art as the universities and the National Science Foundation. The practitioners were so technically minded that they did not find any interest in the uniquely distinctive qualities of the university. The very fact that a good tool is universal means that it does not embody any particular values, any more than a typewriter does-- or even a word processor. The values of things like academic freedom were not immediately essential to producing tools. It was the same basic pattern as compilers and operating systems. If the things worked at all, they put the researcher in the position of doing things which could equally well have been done for a corporation.

It was seemingly impossible to find anything useful to do in academia which a corporation could not do just as well. Another,

and better, solution was to revive engineering radicalism. The idea was to critique corporations, as it were, on their economics instead of their subject matter. The computer science department could do things which were exceedingly practical, but which flew in the face of the economic logic corporations lived by.

Ralph Griswold, who we last saw leaving Stanford to go to Bell Labs, invented the SNOBOL programming language there. In the course of distributing and promoting it, he got into a disagreement with corporate policy. Griswold was acting as if Bell Labs was a national laboratory such as Argonne National Laboratory or Los Alamos, which should operate for the public good. In the case of SNOBOL, this meant placing the software, manuals, etc. in the public domain, so that everyone could use them, and they could become a standard. However corporate management took the view that Bell Labs was a profitmaking corporation, and that software should be sold for a profit. This was of course a shortsighted view for a firm whose finances ultimately depended on public utility franchises, which is to say, on political goodwill, but that was the view of management. In 1971, Griswold left to go to the University of Arizona.

At Arizona, Griswold continued a program of language development and research, setting out to produce his second language, ICON. Griswold obtained National Science Foundation funding, on a modest scale (he estimated $100,000 annually from 1971 to 1990), which was used to pay stipends for graduate students, salaries for laboratory staff, and summer salary for

himself.[1] In other words, he did not ask for enough money to buy the greatest new giant computer, in contrast to the DARPA-oriented labs.

Griswold, in effect, replicated a slice of Bell Labs-- as it ought to have been-- with academic requirements loosely cobbled over the actual organization. As he remarked about dissertation topics:

> Dissertation topics don't come out to be what I thought they might be. And I typically don't assign a topic and say, "Go do this." Nor do I typically expect a student to develop a topic in advance of doing any research and then carry it out. I expect that to develop as a process of learning how to do research and conducting it... If you are committed to doing a big project you just plain have to get it done, and the people that you hire have to be willing to do it and they have to make dissertations out of it somehow - one way or another, which is quite different from truly inspirational things.[2]

The general spirit of this is to make the dissertation into something very like a patent application-- the traditional and authoritative form of engineering writing.

Much the same principle applied to coursework. The students who would eventually come to work with Griswold on ICON tended to get jobs at the campus computer center in the first instance, and then begin taking computer science courses at a leisurely pace. By definition, a promising student was one for whom this level of coursework was essentially recreational. Their real

---

1. BAB OH 201, Ralph Griswold, p. 8, 22.

2. Ibid, pp. 22-23.

challenges would have come in the course of their jobs, and would have involved delving into the internal workings of really large programs written by large numbers of people. Griswold might have had three or four such students at any one time,[1] and his grant money would have sufficed to maintain them at a level of reasonable comfort, so that they did not worry very much about finishing up quickly.

This attitude is illustrated by two of Griswold's students. Stephen Wampler started in the masters program in 1972, switched to the Ph.D program in 1976, completed his Ph.D. in December 1981, and started teaching at Northern Arizona University in January 1982. He had decided that he wanted to teach undergraduates, something he could not conveniently do at the University of Arizona.[2] Presumably, to do that, he needed to get his papers in order. Gregg M. Townsend had gotten a bachelors degree in systems engineering in 1974, and had spent three years working in industry, before coming back to the University of Arizona as a systems programmer at the computer center. He finished up his masters degree in 1984, and shortly afterwards got a job as a Research Programmer in the computer science department, working partly on ICON, and partly on other things. At the time of the interview (1990), Townsend was clear that he did not want to get a Ph.D. He did not want to teach, and he did not particularly care for research per se. What he liked was

---

1. 10-12 Ph.D.'s and 20-30 Masters' over eighteen years, ibid, p. 24.

2. BAB OH 202, Stephen Wampler, pp. 3, 7, 11.

programming, and that was what he did.[1]

Griswold was able to craft a unique identity as a publicly paid developer of public domain software for public use. He recruited students very much like himself, who tended to carry on and expand his program. However, a long-term implication was that it reinforced a tendency for Computer Science to define itself in terms of producing compilers, operating systems, etc.

A large section of Computer Science was driven by the need to give students a good reason not to go off to industry. This meant avoiding policies which led to becoming similar to industry, and at the same time, finding things which industry could not do. After a certain amount of experimentation, the best solution which emerged was to be ethically part of the university, that is, to exemplify the university's values in the new area of computers and software, and to stand in contrast to corporate values. The new computer scientists were in effect betting that sooner or later, AT&T and IBM would revert to type, that is, to the primal capitalism of someone like Jay Gould, or that they would be replaced by someone rather like Jay Gould.

// insert a concluding section

---

1. BAB OH 204, Gregg M. Townsend, pp. 3-4, 7, 10.

An Interview with DALE LAFRENZ, OH 315, Conducted by Judy E. O'Neill on 13 April 1995, Minneapolis MN

An Interview with RALPH and MADGE GRISWOLD, OH 256, Conducted by Judy E. O'Neill on 29 September 1993, Minneapolis, MN

An Interview with RALPH and MADGE GRISWOLD, OH 201, Conducted by David S. Cargo on 25 July 1990, Flagstaff, AZ

An Interview with STEPHEN WAMPLER, OH 202, Conducted by David S. Cargo on 25 July 1990, Flagstaff, AZ

An Interview with GREGG M. TOWNSEND, OH 204, Conducted by David S. Cargo on 26 July 1990, Flagstaff, AZ

An Interview with BRUCE G. BUCHANAN, OH 230, Conducted by Arthur L. Norberg on 11-12 June 1991, Pittsburgh, PA

An Interview with THOMAS A. KEENAN, OH 217, Conducted by William Aspray on 28 September 1990, Washington, D.C.

An Interview with JOSEPH F. TRAUB, OH 70, Conducted by William Aspray on 5 April 1984, Columbia University (New York, NY)

An Interview with GRANGER MORGAN, OH 224, Conducted by Andrew Goldstein on, 27 November 1990, Pittsburgh, PA

An Interview with GENE GOLUB, OH 105, Conducted by Pamela McCorduck on 8 June 1979, San Francisco, CA

An Interview with STEPHEN COOK, OH # 350, Conducted by Philip Frana on 18 October 2002, Toronto, Ontario, Canada

An Interview with TERRY ALLEN WINOGRAD, OH 237, Conducted by Arthur L. Norberg on 11 December 1991, Stanford, CA

An Interview with BERNARD A. GALLER, OH 236, Conducted by Enid H. Galler on 8, 10-11, and 16 August 1991, Sutton's Bay, MI, Ann Arbor, MI

An Interview with WILLIAM F. MILLER, OH 29, Conducted by Pamela McCorduck on 22 May 1979, Stanford, CA

An Interview with JIM GRAY, OH 353, Conducted by Philip L. Frana on 3 January 2002, San Francisco, California

An Interview with CHARLES A. CSURI, OH 180, Conducted by Kerry J. Freedman on 23 October 1989, Columbus, OH

An Interview with FREDERICK WEINGARTEN, OH 212, Conducted by William Aspray on 26 September 1990, Washington, D.C.

James D. Koerner, The Miseducation of American Teachers, Penguin Books, Baltimore, 1965, orig. pub. 1963

Lewis B. Mayhew, "The Future Undergraduate Curriculum," pp. 200-219, in Alvin C. Eurich (The Academy For Educational Development), Campus 1980: The Shape of the Future in American Higher Education, 1968, Dell Publishing, New York.

Jasia Reichardt, The Computer in Art, 1971, Studio Vista, London/Van Nostrand, New York

G. K. Gupta, Computer Science Curriculum Developments in the USA in the 1960's, SCHOOL OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING, MONASH UNIVERSITY (Australia), TECHNICAL REPORT 2004/164, dated 20/12/2004,
http://www.csse.monash.edu.au/
publications/2004/tr-2004-164-abs.html,
http://www.csse.monash.edu.au/
publications/2004/tr-2004-164-full.pdf
downloaded May 9, 2005

Brian Jackson and Dennis Marsden, Education and the Working Class: Some General Themes Raised by a Study of 88 Working-Class Children in a Northern Industrial City, Penguin Books, Ltd., Harmondsworth, Middlesex, England, 1966, orig pub 1962.
Deals with working class enrollment in an academic secondary school (grammar school) in Huddersfield, Yorkshire. Making a somewhat large translation, this would be the analogue of an American state university in terms of social issues.

Hermann Hesse, Magister Ludi (The Glass Bead Game), translated from the German Das Glasperlenspiel by Richard and Clara Winston, with a forward by Theodore Ziolkolski, Bantam Books, New York, 1970, translation orig. pub. 1969, novel orig. pub. 1943.
To get a full sense of Hesse's ideas about the relationship between the intellect and the social world, read also Narcissus and Goldmunde and Beneath the Wheel.

M. Mitchel Waldrop, Complexity: The Emerging Science At the Edge of Order and Chaos, Touchstone (Simon and Schuster), New York, 1993, orig pub 1992.

Forsythe, Stein, Burks, Herriot interviews, Crevier fr. Ch 1

Edwards fr. ch5

Additional sources not yet used:

Alison Adam, "Construction of Gender in the History of Artificial Intelligence," IEEE Annals of the History of Computing, Fall 1996
contains references to various sources for early projects, eg. General Problem Solving System, mostly in the form of

journalistic histories.

The article in the Encyclopedia of Computer Science by N. V. Findler on "Artificial Intelligence" lists assorted items:

Journals:
International Journal of Man-Machine Studies;
LC, from 1969,
Information Sciences;
LC from 1968
International Journal of Computer and Information Sciences;
LC from 1972
Artificial Intelligence;
LC from 1970
Behavioral Science;
LC from 1956
ACM Communications and Journal;
Computer Journal;
Kybernetic; Cybernetica;
IEEE transactions on Computers;
System Science and Cybernetics;
Information and Control; etc.

Books: B. Meltzer and D. Michie, Machine intelligence, 1967-72 (annual workshop proceedings); M. Minsky, Semantic Information Processing, 1968; etc.

Survey Papers, ie. contemporary bibliographies: M. Minsky, "Steps Towards Artificial Intelligence," 1961, Proc. IRE

Other related topics in the Encyclopedia of Computer Science:

The articles in the Encyclopedia of Computer Science on "Arts Applications" and "Humanities Applications" (both by S. A. Sedelow [Sally Yeats Sedelow, U. Kansas]) are another source, since much of this covers the computer as artist. One interesting item is a 1957 article about musical composition by Frederick Brooks, destined to be the "father of the IBM System/360."

Donald G. Fink, Computers and the Human Mind: An Introduction to Artificial Intelligence, 1966 Anchor Books (Doubleday & Company), Science Study Series, Garden City, NY.

The first eight chapters are a necessary introduction to computers, but chapters 9-12 deal with early "hard style" artificial intelligence, eg. Arthur Samuels' checkers player, Newell, Shaw, and Simon, language translation, Hiller's music composition program. This book is significant if nothing else for the sheer number of students who read and were influenced by it. At a time when computers themselves were hard to come by, it was the most accessible material above the level of someone like D. S. Halacy.

Anthony Oettinger's graduate syllabus, and I think, some others, with extensive reading lists in philosophy, psychology, etc.

1. Susumo Kuno and Anthony Oettinger, "Computational Linguistics in a Ph.D. Computer Science Program," CACM, December 1968, pp. 831-36.
2. Robert McNaughton, "Automata, Formal Languages, Abstract Switching, and Computability in an Ph. D. Computer Science Program," CACM, November, 1968, pp. 738-40, 46;
3. Bruce Arden, "The Role of Programming in a Ph.D. Computer Science Program," CACM, January, 1969, pp. 31-37;
4. G. Salton, "Information Science in a Ph. D. Computer Science Program," CACM, February, 1969, pp. 111-17
5. George E. Forsythe, "A University's Educational Program in Computer Science," CACM, January, 1967, pp. 3-11