Making Programming Masculine

Nathan Ensmenger
May 19, 2009

This is the version of the paper presented at the 2009 Conference on Gender and Computing hosted by the Charles Babbage Institute. The final version was published in **Gender Codes: Women and Men in the Computing Professions**, Thomas Misa, ed. (Wiley, 2010)

From Cosmo Girls to Computer Girls

In the April 1967 issue of *Cosmopolitan* magazine, sandwiched between "The Bachelor Girls of Japan" and "A Dog Speaks: Why a Girl Should Own a Pooch," appeared a curious little essay entitled simply "The Computer Girls." As the article explained, these were the female "computer programmers" who taught the dazzling new "miracle machines" called computers "what to do and how to do it." There were already more than 20,000 women working as computer programmers in the United States, argued the article's author, Lois Mandel, and there was an immediate demand for 20,000 more. Not only could a talented "computer girl" command as much $20,000 a year but the opportunities for women in computing were effectively "unlimited." The rapid expansion of the computer industry meant that "sex discrimination in hiring" was unheard of, Mandel confidently declared, and anyone with aptitude—male or female, college-educated or not—could succeed in the field. And not only were women in computing treated as equals, but they actually had many advantages over their male colleagues. Programming was "just like planning a dinner," Mandel quoted the noted computer scientist Dr. Grace Hopper as saying, "You have to plan ahead and schedule everything so it's ready when you need it. Programming requires patience and the ability to handle detail. Women are 'naturals' at computer programming."[1]

It would be easy to dismiss "The Computer Girls" as a fluff piece, a half-hearted attempt by *Cosmopolitan* to capitalize on contemporary interest in the computer revolution. To modern readers the very language of the "computer girl" appears

condescending and sexist. The analogy between computer programming and recipe creation seems forced and superficial. At times the article descends into what seems almost a parody of formulaic *Cosmopolitanism*, such as when Sally Brown, "a redhead from South Bend, Indiana" confesses that she doesn't mind working late because there is often "a nice male programmer to take a girl home …" At one point the author speculates, seemingly without irony, about the "the chances of meeting men in computer work." (The conclusion she comes to is that these are "very good," as the field was currently "overrun" with men.) The last word of the article text comes from a patronizing male programmer: of course "we like having the girls around," he declares, "they're prettier than the rest of us." And, in true *Cosmopolitan* style, the article concludes with a "Cosmo Quiz": by answering a few simple questions, any *Cosmo* girl could see whether she too had what it took to be a professional computer programmer making "$15,000 after five years."[2]

But underneath its seemingly frivolous exterior, "The Computer Girls" gives insight into the gender dynamics of computer work at one of the most critical periods in its history. It reflects very accurately the confusing—and often contradictory—messages about the proper role of women in the computing fields. On the one hand, women did play a critical role in early computing, particularly in computer programming. Compared to most technical professions, computer programming did remain unusually open to females. But on the other hand, in the late 1960s the computer programming community was also actively making itself masculine, pursuing a strategy of professional development that would eventually make it one of the most stereotypically male professions, inhospitable to all but the most adventurous and unconventional women.

Let's begin with the what the *Cosmopolitan* article gets right:

First of all, it is true that in the late 1960s there were an exceptionally large number of women working in computer programming. In fact, if anything the *Cosmo* article *underestimates* the percentage of women programmers. Mandel suggests that one out of every nine working programmers was female. This is probably overly conservative. The exact percentage of female programmers is difficult to pin down with any accuracy—even figuring out the total number of programmers in this period is difficult—but other reliable contemporary observers suggest that it was closer to 30, or even 50, percent.[3] The first government statistics on the programming profession do not appear until 1970, when it was calculated that 22.5 percent of all programmers were women—an estimate more than twice Mandel's.[4]

Of course, computing itself is a very broad term covering a multitude of occupa-

The Computer Girls

BY LOIS MANDEL

A trainee gets $8,000 a year ...a girl "senior systems analyst" gets $20,000—and up! Maybe it's time to investigate....

Ann Richardson, IBM systems engineer, designs a bridge via computer. Above (left) she checks her facts with fellow systems engineer, Marvin V. Fuchs. Right, she feeds facts into the computer. Below, Ann demonstrates on a viewing screen how her facts designed the bridge, and makes changes with a "light pen."

Twenty years ago, a girl could be a secretary, a school teacher . . . maybe a librarian, a social worker or a nurse. If she was really ambitious, she could go into the professions and compete with men . . . usually working harder and longer to earn less pay for the same job.

Now have come the big, dazzling computers—and a whole new kind of work for women: programming. Telling the miracle machines what to do and how to do it. Anything from predicting the weather to sending out billing notices from the local department store.

And if it doesn't sound like woman's work—well, it just is.

("I had this idea I'd be standing at a big machine and pressing buttons all day long," says a girl who programs for a Los Angeles bank. I couldn't have been further off the track. I figure out how the computer can solve a problem, and then instruct the machine to do it."

"It's just like planning a dinner," explains Dr. Grace Hopper, now a staff scientist in systems programming for Univac. (She helped develop the first electronic digital computer, the Eniac, in 1946.) "You have to plan ahead and schedule everything so it's ready when you need it. Programming requires patience and the ability to handle detail. Women are 'naturals' at computer programming."

What she's talking about is *aptitude*—the one most important quality a girl needs to become a programmer. She also needs a keen, logical mind. And if that zeroes out the old Billie Burke-Gracie Allen image of femininity, it's about time, because this is the age of the Computer Girls. There are twenty thousand of them in the United *(cont. on page 54)*

Photos by Henry Grossman. Dress by Gino Charles.

Figure 1: The Computer Girls, *Cosmopolitan*, April 1967

tional categories, including high-status jobs like computer programming and systems analysis as well as low-status jobs such as keypunch operator. Women tended to congregate in the lower end of the occupational pool in computing. Even within computer programming there were different roles differentially available to men and women. But as the *Cosmopolitan* article rightly points out, compared to most of the traditional professions, computer programming was remarkably receptive to females. One of the programmers it profiled, Helene Carson, had previously earned an MA degree in astrophysics at Harvard. Although Carlson had discovered that "there wasn't much a woman could do in astronomy," in computing she felt that she had been "fully accepted as a professional."[5]

Again, there is evidence that Carlson (and *Cosmo*) was absolutely spot-on in regard to the vertical mobility available to women in computer programming. Compared to other technical disciplines, computer programming was not highly stratified along gender lines. Not only were women able to break into the entry levels of the profession, but some were able to climb to its highest pinnacles. In 1969, for example, the Data Processing Management Association recognized Grace Hopper with its very first "man of the year" award in the computer sciences. That an emerging professional society with grand aspirations for technical and managerial leadership would even consider giving its *first* major award to a woman is really quite remarkable. Although Hopper was unusual in that she possessed both a Ph.D. and a commission in the United States Navy (at that time as a Lieutenant Commander), she was not entirely *sui generis*: other women, including Betty Snyder Holberton, Jean Sammet, and Beatrice Helen Worsley, all came to occupy influential positions within the computing community.[6]

In addition to accurately representing the state of the contemporary labor market in programming, the *Cosmopolitan* article also does a reasonable job of explaining its unique characteristics. In large part, the unusual freedom of opportunity available to women in computing was simply an outgrowth of the rapid growth of the commercial computer industry. An industry that was doubling in size every year or two simply could not afford to discriminate against women. "*Every* company that makes or uses computers hires women to program them," the article noted matter-of-factly, "If a girl is qualified, she's got the job." And since the meaning of "qualified" in this period was still being negotiated (more on this point later), there was no particular reason for firms to privilege men over women.[7]

It would be difficult to over-emphasize the degree to which the programmer labor shortage of the 1960s dominated contemporary discussions of the health and future of the computer industry. For years, industry employers had been warning of an imminent shortage of computer programmers. The "gap in programming

support" threatened to wreak havoc with the industry.[8] In 1962 the editors of the powerful industry journal *Datamation* declared that "first on anyone's checklist of professional problems is the manpower shortage of both trained and even untrained programmers, operators, logical designers and engineers."[9] In 1966 the "personnel crisis" had developed into a full blown "software crisis," according to *Business Week* magazine.[10] An informal 1967 survey of MIS (management information systems) managers identified as the primary hurdle "handicapping the progress of MIS" to be "the shortage of good, experienced people."[11] One widely quoted study released that same year noted that although there were already 100,000 programmers working in the United States, there was an immediate need for at least 50,000 more.[12] Estimates of the number of programmers that would be required by 1970 ranged as high as 650,000.[13] "Competition for programmers has driven salaries up so fast," warned *Fortune* in 1967, "that programming has become probably the country's highest paid technological occupation …Even so, some companies can't find experienced programmers at any price."[14] The ongoing "shortage of capable programmers," argued *Datamation*, "had profound implications, not only for the computer industry as it is now, but for how it can be in the future."[15]

In the face of this perpetual shortage of programmers, employers turned to extraordinary measures. Recruitment companies scoured local community centers and YMCAs for potential programmer trainees, administering programming aptitude tests to almost every warm body they could find.[16] In 1968 one computer service bureau in New York City even began testing inmates at the nearby Sing-Sing prison, promising them permanent positions on their release.[17] Given that employers were willing to hire prisoners as programmers, their appeal to *Cosmopolitan* readers is unexceptional. As in the case of other severe labor shortages—wartime, for example—women were able to move into fields from which they might otherwise have been excluded.

The combination of low-barriers to entry and subsidized technical education made programming powerfully appealing to many women who might otherwise be trapped in traditionally female occupations. But it was not only the desperate need for programmers that allowed women unique opportunities within the profession. Although in the late 1960s programming was generally considered highly-skilled labor—as one observer declared, "generating software is 'brain business,' often an agonizingly difficult intellectual effort"—the exact nature of that intellectual effort was not yet clearly defined.[18] Programming was "not yet a science," argued the same observer, "but an art that lacks standards, definitions, agreement on theories and approaches."[19] The lack of a fully established scientific or engineering identity left space open for women. Although the possession of a college degree in mathematics

was still considered a necessity in scientific computing (which tipped the scales demographically in favor of males), business computing—the most rapidly growing segment of the commercial computer programming industry—required an entirely different set of skills. What these skills were no-one quite knew, and so many firms relied on aptitude tests to determine which employees had the most potential for programming. Aptitude was everything; you either had it or you didn't. And since there was no particular reason that these aptitude tests were gender-specific (again, more on this later), there was also no reason that men would be more likely than women to be selected as programmer trainees. In addition, as the *Cosmo* article also correctly noted, since most firms preferred to train programmers from within, and therefore often tested *all* of their employees for programming aptitude, even women working in such highly feminized (and low-status) occupations as stenography had a chance at becoming a programmer. The trick was getting some initial experience: as one employment counselor cited by Mandel argued, "a girl's best bet is to get a spot *anywhere* in the computer department, using skills like filing or typing or accounting, with the plan in mind to get on the firm's programmer-trainee list from the inside." There were outside vocational schools that claimed to prepare people for careers in programming, but as one of the "girls" quoted in the article declared, "I'd never consider paying for my own training when I can get someone else to pay for it."[20]

It is worth noting as well that, given this context, the quiz provided at the end of "The Computer Girls" article was no superfluous or silly afterthought. The quiz included real questions from the aptitude test developed by NCR to test for programming aptitude. Similar tests, most notably the IBM Programmer Aptitude Test (PAT), were used by eighty percent of all employers to select for programmer trainees.[21] In 1967 alone, the PAT was administered to more than 700,000 individuals.[22]

In any case, after noting a few other reasons why programming might be an appealing profession for women—including that at least some programming work could be done at home (while children were napping)—the *Cosmo* article concluded by suggesting that it was largely a lack of knowledge about the field that kept women from entering it in greater numbers. Since programming was thought to be vaguely mathematical in nature (incorrectly, the article concludes), and since female students were often discouraged from pursuing any fields involving science or mathematics, they too often missed out on the exciting opportunities available in programming. This was unfortunate. "I don't know of any other field, outside of teaching, where there's as much opportunity for a woman," the article quoted the director of education for the Association for Computing Machinery, James Adams, as saying. "Soon, mothers will be telling their daughters: 'Now study your arithmetic so you can become a computer girl.'"

What makes the vision of widespread female participation in the computer industry portrayed in "The Computer Girls" so intriguing today, of course, is that it is so unfamiliar. From a contemporary perspective, the computing professions appear egregiously male-dominated. The problem of female participation in computer science programs—declining since the mid-1980s—is of particular concern, and is generally explained in terms of "opening up" the discipline to women. The idea that many of the computing professions were not only historically unusually accepting of women, but were in fact were once considered "feminized" occupations, seems extraordinary, if not unbelievable. And yet an historical understanding of how the computing professions acquired their gendered identity, how they were "made masculine" is critical to any attempt to address the current gender imbalance in computing. The historical perspective, in this case, is not only relevant, but essential.

Beginning in the 1990s, historians of computing began to recognize the crucial contributions that women have made to the development of electronic computing. Like many such (re)discoveries of the previously unrecognized contributions of women, this one had both historical and contemporary significance. Given that computing was generally considered to be particularly masculine (even when compared to the traditionally male-dominated engineering disciplines), the surprisingly large presence of women in early computing seemed to turn on its head conventional assumptions about the lack of female participation in contemporary computing. It wasn't that women were uninterested in computing, or unprepared or constitutionally disinclined to participate, the historical evidence seemed to suggest, but rather that their participation had been systematically ignored or under-reported.[23] In light of contemporary debates about low (and declining) female enrollments in departments of computer science, this seemed a significant and empowering discovery.[24] The focus of most of this literature has been, understandably enough, on what Judy Wacjman, among others, has called the "hidden history" of women in technology.[25] The goal was to explore what the history of women in computing had to say about women—about their contributions, experiences, and abilities.[26]

This chapter will address instead the flip side of this question: namely, what the history of women in computing has to say about *computing*. Because of the modern association of computer work—particularly computer programming—with high-status males, we tend to assume that such work has always been masculine, and that the presence of women is therefore exceptional. My argument is that most computer work—again, particularly computer programming—began as women's work. It had to be made masculine. This process of masculinization was closely associated with the development of the professional structures of the discipline: formal programs in computer science, professional journals and societies, certification programs and

standardized development methodologies. Seen from the perspective of aspiring computer professionals (primarily male), "The Computer Girls" represented not a celebration of the openness and opportunity inherent in their industry, but an indictment of everything that was wrong with it.

In terms of the larger questions addressed in this volume, this chapter provides important insights into the way in which the *structures* of a profession both reflect and replicate the *culture* of its practitioners. One of the most significant barriers to female participation in computing is the culture of computing, a culture which is perceived to be inherently (and excessively) masculine. The roots of this culture reach back into the early history of electronic computing, and can only be understood, and addressed, in the context of a full historical understanding of its origins.

In The Beginning Were The Women ...

The most prominent case study in the history of women in early computing is, in fact, the earliest. In the early 1940s a group of six women—Kathleen McNulty, Frances Bilas, Betty Jean Jennings, Betty Holberton, Ruth Lichterman, and Marlyn Wescoff—were recruited to assist with the development and operation of the University of Pennsylvania's ENIAC machine. The ENIAC (Electronic Numerical Integrator And Computer) was one of the first, and certainly most famous, early electronic computers, and the "ENIAC Girls" (as they were often referred to by contemporaries) were the female "human computers" recruited by the male ENIAC engineers/managers to "set-up" the general-purpose ENIAC machine to perform the specific "plans of computation" required to solve real-world problems. Although the idea of the computer "program" had not yet been developed, the women of ENIAC are nevertheless widely celebrated as the world's first computer programmers. And not only was the pioneering work that they did on the ENIAC historically significant, many went on to subsequent careers—often at the highest levels—in electronic computing.

The expectation was that the work of "setting up" the ENIAC would be relatively trivial. But in his 1996 article based on interviews with the ENIAC programmers, Barkley Fritz highlights the substantial contributions that these women made to the operation—and particularly the troubleshooting—of the ENIAC. According to Betty Jean Jennings, for example, the ENIAC women learned to understand the internal wiring diagrams of the ENIAC machine, and "as a result we could diagnose troubles almost down to the individual vacuum tube. Since we knew both the application and the machine, we learned to diagnose troubles as well as, if not better than, the engineer."[27] In a few cases these female programmers significantly affected

the design of the ENIAC and subsequent computers. ENIAC programmer Betty Holberton recalled one particularly significant episode when she convinced John von Neumann to include a "stop instruction" in the machine: although initially dismissive, von Neumann eventually recognized the programmer's legitimate need for such an instruction. Other accounts by participants and observers echo the critically important—but generally unanticipated—role that the ENIAC programmers played in facilitating the successful launch of one of the world's most famous early electronic computers. Yet, as Jennifer Light has convincingly demonstrated, the contributions of these women were subsequently systematically eliminated from the historical record.[28]

There is no question that the work of the ENIAC women was disregarded in large part simply because they were women. But almost as significant as their gender was their subordinate position as "software" workers in a hardware oriented development project. Obviously the two are closely related. Of course, the use of the word "software" in this context is anachronistic—the word itself would not be introduced until 1958—but the hierarchical distinctions and gender connotations it embodies—between "hard" technical mastery and the "softer," more social (and implicitly, of secondary importance) aspects of computer work—are applicable.[29] In the status hierarchy of the ENIAC project, it was clearly the male computer engineers who were significant. The ENIAC women were expected to simply adapt the "plans of computation" already widely used in human computing projects to the new technology of the electronic computer. These "plans of computation" were themselves highly gendered, having been traditionally developed by women for women (human computing had been largely feminized by the 1940s). The ENIAC women would simply "set-up" the machine to perform these pre-determined plans: that this work would, in fact, be difficult and require radically innovative thinking was completely unanticipated.[30] The telephone switchboard-like appearance of the ENIAC programming cable-and-plug panels reinforced the notion that programmers were mere machine operators, that programming was more handicraft than science, more feminine than masculine, more mechanical than intellectual.

The idea that the development of hardware was the real business of computing, and that software was at best secondary, persisted for many years. In the first textbook on computing published in the United States, for example, John von Neumann and Herman Goldstine outlined a clear division of labor in computing—presumably based on their experience with the ENIAC project—that clearly distinguished between the "head-work" of the (male) scientist, or "planner," and "hand-work" of the (largely female) "coder."[31] In the Goldstine/von Neumann schema, the "planner" did the intellectual work of analysis, and the "coder" simply translated this work into a form

that a computer could understand. "Coding" was a "static" process that could be performed by a low-level of clerical worker. "Coding" implied mechanical translation or rote transcription; "coders" were obviously low on the intellectual and professional status hierarchy. It was not unreasonable to expect that, as was the case in the ENIAC project, that most of these "coders" would be women.

An early manuscript version of the UNIVAC "Introduction to Programming" manual mirrored this distinction between "planner" and "coder." In this instance the term "programmer" was used, somewhat unconventionally, in place of "planner", but the distinction between the analytical "programmer" (the person who "studies the problem, determines the appropriate method of solution, and prepares the flow chart") and the clerical "coder" (who "need only be familiar with the technique of reducing the flow chart to the specific instructions, or coding, required by the UNIVAC to solve the problem") remains the same.[32] In the UNIVAC manual, like the Goldstine/von Neumann textbook, the real business of programming was analysis: the actual coding aspect of programming was trivial and mechanical.

It was not until the early 1950s that the term "programmer" was widely adopted within the computing community. As David Grier has suggested, the verb "to program," with its military connotations of "to assemble" or "to organize," suggested a more thoughtful and system oriented activity.[33] But even as "programmer" was increasingly adopted within the computing community, software workers would struggle to distance themselves from the status (and gender) connotations suggested by the older designation "coder." The accusation that programmers were "mere coders" was used throughout the 1950s and 1960s by those who wanted to counter the influence of "uppity" software workers. The noted computer scientist John Backus, for example, argued that the adoption of the title "programmer" by former "coders" happened "for the same reason that janitors are now called 'custodians'": "Programmer was considered a higher class enterprise than 'coder,' and things have a tendency to move in that direction."[34]

The conflation of programming and coding, and the association of both with low-status clerical labor, suggested the ways in which early software workers were gendered female. In the ENIAC project, of course, the programmers actually were women. But the suggestion that "coding" was low-status clerical work also implied an additional association with female labor. As Majory Davies, Sharon Strom, and Elyce Rotella have described, clerical work had, by the second decade of the 20th century, become largely feminized.[35] This was particularly true of clerical occupations that were characterized by the rigid division of labor and the introduction of new technologies. Some of these occupations carried over directly into the computer era: the job of keypunch operator, for example, had been thoroughly feminized long before

it became associated with electronic data processing.[36] And although today we do not associate the work of key-punchers with the work of the computer programmer, in the 1950s and 1960s the differentiation between keypunch operator and other forms of computer work was not always clear. The *Cosmopolitan* article, for example, lumped keypunch operators in among the "computer girls," and other contemporary sources identified keypunch operators as an obvious source of programmer trainees.[37] In any case, the historical pattern has been that low-status occupations, with the exception of those requiring certain forms of physical strength, have often become feminized.

The "Bad Boys" of Programming

In the 1950s, however, computer programming was beginning to acquire new status and a new gender identity. The experience of the ENIAC girls had shown that electronic computing was anything but an "automated form of hand computation." The neat distinction made by Goldstine and von Neumann between analysis and implementation quickly broke down in practice. To begin with, since the primary purpose of the earliest computers was to produce solutions to complex mathematical functions that could not be solved analytically, the programmers of these computers necessarily required skill in numerical analysis. This process of analysis was itself something of an art form: numerical solutions always involved a compromise between speed and accuracy—even when using the fastest computers. Choosing the right approximation involved balancing acceptable error against the specific limitations of a given machine—a process that required daring, creativity, and mathematical intuition.

Perhaps even more significantly, the performance and memory constraints of the first generation of electronic computers demanded that programmers cultivate a series of idiosyncratic craft techniques to overcome the limitations of primitive hardware. For example, contemporary memory devices were so slow and had such little capacity that programmers had to develop ingenious techniques to fit their programs into the available memory space. In order to coax every bit of speed out of a relatively slow storage device such as a rotating memory drum, programmers would carefully organize their coded instructions in such a way as to assure that the each instruction passed by the magnetic read head in the right order and at just the right execution time. Only the best programmers could hope to develop applications that worked at acceptable levels of usability and performance.

For all of these reasons, programming began to acquire a reputation for being incomprehensible to all but a small set of extremely talented insiders. As John

Backus would later describe it, "programming in the 1950s was a black art, a private arcane matter …each problem required a unique beginning at square one, and the success of a program depended primarily on the programmer's private techniques and invention." Techniques developed for one application or installation could not be easily adapted for other purposes. There were few useful or widely applicable tools available to programmers, and certainly no "science" of programming. Programmers often worked in relative isolation, and had few opportunities for formal or even informal education. They generally perceived little value in the work going on at other firms or laboratories, as it was equally haphazard and idiosyncratic. They placed great emphasis on local knowledge and individual ability.

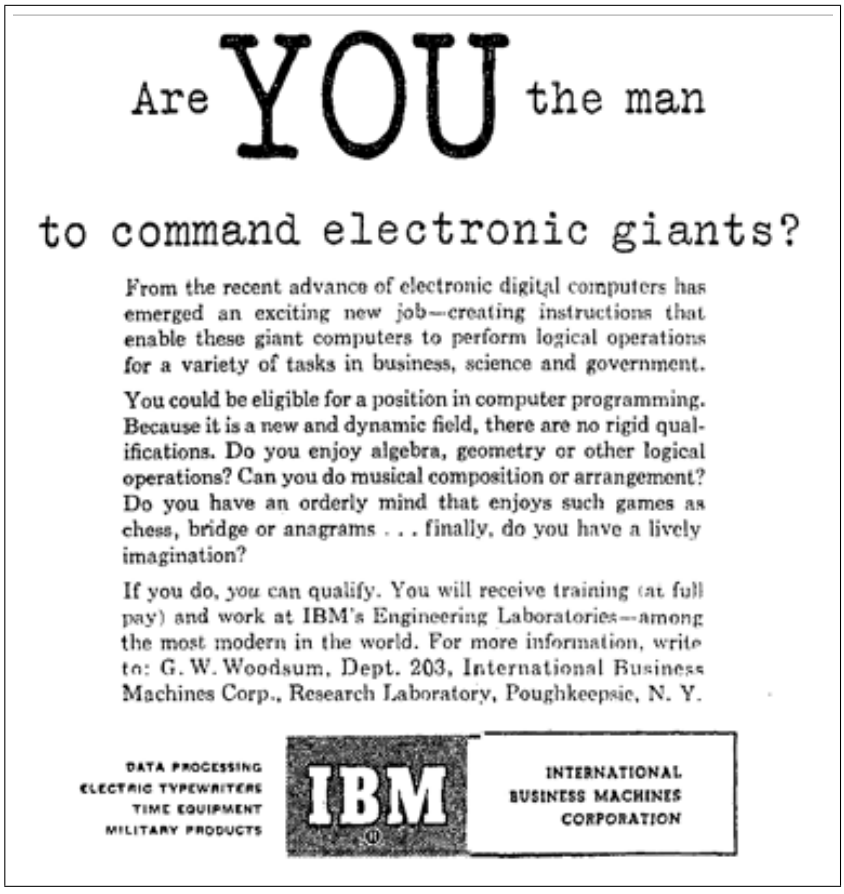

Figure 2: IBM Advertisement, *The New York Times*, May 13, 1956

The heady combination of mathematics, engineering "tinkering," and arcane technique attracted a certain kind of male to computer programming. Some had

abandoned careers in more established scientific disciplines to pursue the emerging field of electronic computing. Others drifted in from mathematics or electrical engineering, or from careers in business or data processing. A few, such as the physicist-turned-programmer Edsger Dijkstra, worried about the lack of a "sound body of knowledge that could support it [programming] as an intellectually respectable discipline."[38] The popular notion that programmers were idiosyncratic geniuses, and that "a really competent programmer should be puzzle-minded and very fond of clever tricks" was a pernicious anachronism, Dijkstra would later argue, that encouraged a short-sighted, "tinkering" approach to software development. Academically-minded programmers like Dijkstra felt that too many of their colleagues regarded their work as temporary solutions to local problems, rather than as an opportunity to develop a more permanent body of knowledge and technique. What computing needed to realize its true revolutionary potential, Dijkstra argued, was a more rigorous approach to programming, one modeled after the science of applied mathematics.[39] But most programmers accepted—and many reveled in—the conventional belief that, at least for the conceivable future, programming would remain the exclusive domain of the select few who possessed the "right stuff." Either way, this new occupational and professional identity, whether based on the academic prestige of the emerging discipline of computer science, or the exclusivity of the "lone gun" tinkerer, was essentially masculine.

This perception of programming as an idiosyncratic arcane discipline—and by extension, its practitioners a "long-haired programming priesthood" [40]—was reinforced by a series of aptitude tests and personality profiles that suggested that focused on innate abilities. By the mid-1960s the majority of companies (80%) were using such tests and profiles as their primary tool for identifying programmer-trainees. "Creativity is a major attribute of technically oriented people," suggested one representative profile: "Look for those who like intellectual challenge rather than interpersonal relations or managerial decision-making. Look for the chess player, the solver of mathematical puzzles."[41] Many of the advertisements for programmers in this period specifically reference chess-playing, musical ability, and mathematics.[42] In 1956 IBM launched an advertisement for programmers that led to the hiring of such notable chessmen as Arthur Bisquier, the U.S. Open Chess champion, Alex Bernstein, a U.S. Collegiate champion, and Sid Noble, the self-proclaimed "chess champion of the French Riviera."[43] (It should be noted, however, that the same campaign also netted an Oxford trained crystallographer, an English Ph.D. candidate from Columbia University, an ex-fashion model [female], and a "proto-hippie," so obviously chess-playing ability was not the sole criteria.) In any case, good programming was believed to be dependent on uniquely qualified individuals, and that what defined these uniquely

individuals was some indescribable, impalpable quality—a "twinkle in the eye," an "indefinable enthusiasm," or what one interviewer described as "the programming bug that meant … we're going to take a chance on him despite his background."[44]

In addition, great disparities were discovered between the productivity of individual programmers, with one widely cited IBM study suggesting that a truly excellent programmer was twenty-six times more efficient than his merely average colleagues.[45] Despite the serious methodological flaws that compromised this particular study (including a sample population of only twelve individuals), the 26:1 performance ratio quickly became part of the standard lore of the industry. "When a programmer is good, he is very, very good. But when he is bad, he is horrid," the study declared, reinforcing the notion that skilled programmers were thought to be effectively irreplaceable, and were to be treated and compensated accordingly. Programmers were to be selected for their intellectual gifts and aptitudes, rather than their business knowledge or managerial savvy.

The notion that programming was a "black art" pervades the literature from the early decades of computing. Even today, more than a half-century after the invention of the first electronic computers, the notion that the computer programming still retains an essentially "artistic" character is still widely accepted.[46] Whether or not this is true or desirable is an entirely different question—a subject of considerable and contentious debate. What is important is that by characterizing the work that they did as "artistic," programmers could lay claim to the autonomy and authority that came with being an artist. Note that the appeal here is to the tradition of the artisan, or craftsman, which is a masculine identity, not the potentially effeminate "artsy" type.

The widespread perception that programming ability was an innate ability, rather than an acquired skill or the product of a particular form of technical education, could be seen as gender-neutral, or even female-friendly. After all, the aptitude tests for programming ability were widely distributed among female employees, including clerical workers and secretaries. And, according to one 1968 study, it was found that a successful team of computer specialists included an "ex-farmer, a former tabulating machine operator, an ex-key punch operator, a girl who had done secretarial work, a musician and a graduate in mathematics." Of these, the mathematician "was considered the least competent."[47] As hiring practices went, aptitude testing at least had the virtue of being impersonal and seemingly objective. Being a member of the "old boys club" does not do much for one's scores on a standardized exam.[48]

But the aptitude tests and personality profiles did embody and privilege masculine characteristics. For example, despite the growing consensus within the industry (particularly in business data processing) that mathematical training was irrelevant

to most commercial programming, popular aptitude tests such as the IBM PAT still emphasized mathematical ability.[49] Some of the mathematical questions tested only logical thinking and pattern recognition, but others required formal training in mathematics—a fact that *Cosmopolitan* noted as discriminating against women.

Even worse were the personality profiles. The use of personality profiles to identify programmers began, as with other industry-standard recruiting practices, at the System Development Corporation (SDC), the RAND Corporation spin-off charged with the development of the software for the SAGE air-defense system. Faced with the need to train computer programmers in unprecedented numbers—in 1956 SDC employed 700 programmers, almost three-fifths of the total number of programmers available world-wide, and by the beginning of the 1960s had trained 7,000 more—SDC relied extensively on aptitude testing and personality profiling. By the beginning of the 1960s, however, SDC psychologists had developed more sophisticated models based on the extensive employment data the company had collected over the previous decade, as well as surveys of members of the Association for Computing Machinery and the Data Processing Management Association. In a series of papers published in serious academic journals such as the *Journal of Applied Psychology* and *Personnel Psychology*, SDC psychologists Dallis Perry and William Cannon provided a detailed profile of the "vocational interests of computer programmers."[50] The scientific basis for their profile was the Strong Vocational Interest Bank (SVIB), which had been widely used in vocational testing since the late 1920s.

The basic SVIB in this period consisted of four hundred questions aimed at eliciting an emotional response ("like", "dislike," or "indifferent") to specific occupations, work and recreational activities, types of people, and personality types. By the 1960s, more than fifty statistically significant collections of preferences ("keys") had been developed for such occupations as artist, mathematician, policeman, and airplane pilot. The assumption behind the use of such profiles was that candidates who had interests in common with those individuals who were successful in a given occupation were themselves also likely to achieve similar success.

Many of the traits that Perry and Cannon attributed to successful programmers were unremarkable: for the most part programmers enjoyed their work, disliked routine and regimentation, and were especially interested in problem and puzzle-solving activities.[51] The programmer key they developed bore some resemblance to the existing keys for engineering and chemistry, but not to those of physics or mathematics, which Perry and Cannon interpreted as a refutation of the traditional focus on mathematics training in programmer recruitment. Otherwise, programmers resembled other white-collar professionals in such diverse fields as optometry, public administration, accounting, and personnel management.

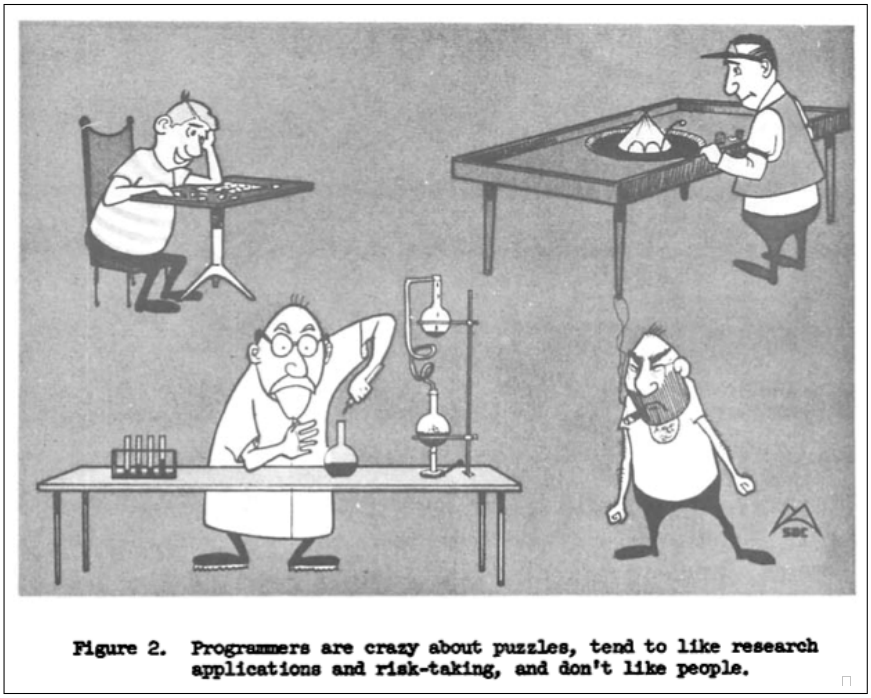

Figure 2. Programmers are crazy about puzzles, tend to like research applications and risk-taking, and don't like people.

Figure 3: William Cannon, *A vocational interest scale for computer programmers* (1966)

In fact, there was only one really "striking characteristic" about programmers that the Perry and Cannon study identified. This was "their disinterest in people." Compared with other professional men, "programmers dislike activities involving close personal interaction. They prefer to work with things rather than people."[52] In a subsequent study, Perry and Cannon demonstrated this to be true of female programmers as well.[53]

The idea that computer programmers lacked "people skills" quickly became part of the lore of the computer industry. The influential industry analyst Richard Brandon argued that this was in part a reflection of the selection process itself, with its emphasis on mathematics and logic. The "Darwinian selection" mechanism of personnel profiling, Brandon suggested, selected for personality traits that performed well in the artificial isolation of the testing environment, but which proved dysfunctional in the more complex social environment of a corporate development project. Programmers were "excessively independent," argued Brandon, often to the point of mild paranoia. The programmer type is "often egocentric, slightly neurotic, and he borders upon a limited schizophrenia. The incidence of beards, sandals, and other symptoms of rugged individualism or nonconformity are notably greater among this demographic group. Stories about programmers and their attitudes and peculiarities are legion, and do not bear repeating here."[54]

Needless to say, these psychological profiles embodied a preference for stereotypically masculine characteristics. A 1970 review of the psychometric literature noted that computer programmers received unusually high masculinity and low femininity scores. In fact, only four occupational groups received higher masculinity scores (unfortunately, the review does not mention which four). "These consistent results [high masculinity scores] define one characteristic of the people in data processing jobs," the review concluded—namely, their masculine self-identity.[55]

The idea that "detached" (read male) individuals made good programmers was embodied, in the form of the psychological profile, into the hiring practices of the industry.[56] Possibly this was a legacy of the murky origins of programming in the early 1950s; perhaps it was self-fulfilling prophecy. Nevertheless, the idea of the programmer as being particularly ill-equipped for or uninterested in social interaction did become part of the conventional wisdom of the industry. The association of masculine personality characteristics with inherent programming ability helped created an occupational culture in which female programmers were seen as exceptional or marginal. Only by behaving less "female," could they be perceived as being acceptable. Many women still did continue to be hired as programmers and other computer specialists, but they did so in an environment that was becoming increasingly normalized as masculine.

Figure 4: Datamation Cartoon, 1963

One interpretation of the male bias embedded in these aptitude tests and personality profiles is that such tests are, in fact, an accurate reflection of the mental or emotional characteristics that make for a good programmer—logical, detached, anti-social—and that these traits just happen to be more predominant in males. This is the essentialist argument: gender discrimination as a function of biology. Even in the 1960s and 1970s there seemed little evidence for such reductionist explanations.[57]

A second interpretation is that the tests were developed deliberately to exclude women from an increasingly high-status, lucrative, and therefore male-dominated profession. This is the conspiratorial argument.

Another interpretation is that programming ability has no correlation at all with biologically-determined predispositions, but that the widespread use of gender-biased testing regimes by industry employers nevertheless did create a feedback cycle that ultimately selected for programmers with stereotypically masculine characteristics. The primary selection mechanism used by the industry selected for anti-social, mathematically inclined males, and therefore anti-social, mathematically inclined males were over-represented in the programmer population; this in turn reinforced the popular perception that programmers *ought* to be anti-social and mathematically-inclined (and male), and so on *ad infinitum*. This would be an historically continent argument: gender discrimination as a function of historical accident.

It is this last explanation that seems most plausible. In the case of aptitude testing and personality profiling, at least, it appears that the privileging of masculine characteristics is the result of some combination of laziness, ambiguity, and traditional male privilege. There was widespread evidence, even in the late 1960s, that psychometric testing was inaccurate, unscientific, had been widely compromised, and was a poor predictor of future performance. Nevertheless, these methods continued to be used simply because they were convenient. The rapid expansion of the commercial computer industry in the early 1960s demanded the recruitment of large armies of new professional programmers. At the same time, the general lack of consensus about what constituted relevant knowledge or experience in the computer fields undermined attempts to systematize the production of programmers. Commercial programming schools were seen as being too lax in their standards; the emerging academic discipline of computer science, too stringent. Neither offered a reliable short-term solution to the burgeoning labor shortage in programming. In the face of such uncertainty and ambiguity, aptitude testing and personality profiling promised at least the illusion of managerial control. To borrow a phrase from contemporary computer industry parlance, aptitude testing was a solution that *scaled efficiently*. That is to say, the costs of aptitude testing grew in a predictable, linear relationship to the number of applicants (as opposed to other recruitment methods such as personal

interviews, whose costs in time and money grew rapidly.) Put even more simply, it was possible to administer aptitude tests quickly and inexpensively to thousands of aspiring programmers. Compared to its time-consuming and expensive alternatives, aptitude testing was a cheap and easy solution. And since the contemporary emphasis on individual genius over experience or education meant that a star programmer was as likely to come from the secretarial pool as the engineering department, the ability to screen large numbers of potential trainees was preeminent.

But the kinds of questions that could be easily tested using multiple choice aptitude tests and mass-administered personality profiles necessarily focused on mathematical trivia, logic puzzles, and word games. The test format simply did not allow for any more nuanced or meaningful or context-specific problem solving. And, in the 1950s and 1960s at least, such questions did privilege the typical male educational experience. Again, this bias towards male programmers was not so much deliberate as it was convenient. The fact that the use of lazy screening practices inadvertently excluded large number of potential *female* trainees was simply never considered. But the increasing assumption that the average programmer was also male did play a key role in the establishment of a highly masculine programming subculture.

There has been much written in recent years about the distinctively masculine culture of computing and the way in this culture discourages women from entering the computing professions.[58] Of all of the explanations given for the deplorably low rates of female participation in computing (or at least in academic computer science), cultural arguments are the most convincing. It is important to note, therefore, that the origins of this culture lie not in the early 1940s, with the invention of computing, but in subsequent decades. This culture was not inherent in electronic computing, or even adopted directly from related disciplines; it had to be created, and recreated, over the course of decades. One of the essential ways in which this culture was replicated was through the development of practices and institutions.

Professionalization = Masculinization

The process of making programming masculine did not begin—or end—with the transformation of the feminized clerical work of "coding" into the highly masculine, seat-of-the-pants "black art" of programming of the 1950s, not even with the embodiment of certain masculine values into the hiring procedures of the industry. To begin with, this transformation was never fully complete. Aspects of programming remained rote, mechanical, and low-status. It was also not clear that the frontier mentality of programming culture in the 1950s was anything but a function of the immaturity of the industry. The influx of new programmer trainees and vocational

school graduates into the software labor market exacerbated an already bad labor situation. The market was flooded with aspiring programmers with little training and no practical experience. As one study by the Association for Computing Machinery's Special Interest Group on Computer Personnel Research (SIGCPR) warned, by 1968 there was a growing *oversupply* of a certain undesirable species of software specialist. "The ranks of the computer world are being swelled by growing hordes of programmers, systems analysts and related personnel," the SIGCPR argued. "Educational, performance and professional standards are virtually nonexistent and confusion grows rampant in selecting, training, and assigning people to do jobs."[59] At the same time that the demand for skilled programmers was increasing dramatically (and seemingly without limit), when salaries and opportunities for occupational mobility were at their peak, many programmers were plagued with uncertainty about the status and future of their discipline.

There were tangible reasons for this uncertainty. The increasing capabilities and reliability of second generation hardware meant that the baroque "work arounds" and optimizations so prized by programmer-tinkerers were no longer necessary. In addition, the development of so-called "automatic programming systems" threatened to make programmers obsolete altogether, and to return responsibility for the "head work" involved in problem analysis back to the scientists and managers. The persistent lack of programmers to develop a "scientific" basis for their discipline suggested that they were at best artisans or technicians, the last vestiges of a "pre-industrial" approach to software development.[60] The organizational tensions provoked by the increasing use of computerized systems for managerial purposes created resentment against the perceived "abdication" of management imperatives to whiz-kid "computer boys."[61] These tensions reflected themselves in active attempts by managers to reassert their traditional authority over computer programmers by redefining their work as "merely" technical. Finally, the rising cost of software relative to hardware meant that firms began looking for ways to reduce costs by "rationalizing" their development practices. Such "rationalization" often meant the incorporation of a less-expensive, lower-skill (read feminized) work force.

Certainly corporations, academics, and other reformers tried to rationalize the practices of computer programmers in response to the emerging "software crisis" of the late 1960s. In *Programmers and Managers: The Routinization of Computer Programming in the United States*, the historian Philip Kraft argued that managers had, in fact, been successful in "degrading" the work of computer specialists. "Programmers, systems analysts, and other software workers," he argued, were the victims of efforts to "break down, simplify, routinize, and standardize" their work practices. Kraft suggested that corporate managers had generally been successful in imposing

structures on programmers that have eliminated their creativity and autonomy. His analysis was remarkably comprehensive, covering such issues as training and education, structured programming techniques ("the software manager's answer to the conveyor belt"), the social organization of the workplace (aimed at reinforcing the fragmentation between "head" planning and "hand" labor), and careers, pay, and professionalism (encouraged by managers as a means of discouraging unions). In 1979 Joan Greenbaum echoed Kraft's conclusions, arguing that "If we strip away the spin words used today like 'knowledge' worker, 'flexible' work, and 'high tech' work, and if we insert the word 'information system' for 'machinery,' we are still talking about management attempts to control and coordinate labor processes."[62] More recently, Greg Downey has suggested a connection between routinization, feminization, and the increasing of foreign labor in software development ("outsourcing").[63]

It is questionable how successful corporate managers and other "rationalizers" were in their quest to transform software development into a controlled, industrial manufacturing process. Computer programmers are, on the whole, well-paid, highly-valued, and largely autonomous professionals. But it is clear that many programmers in the 1960s were worried about the *possibility* of having their work routinized and degraded. Certainly the management literature from this period is full of confident claims about the ability of new performance metrics, development methodologies, and automatic programming languages to reduce corporate dependence on individual programmers.[64] As Michael Cusamano has described, the vision of the "software factory"—in which hordes of low-paid, low-skill programmers cranked out mass-produced software products—was a persistent theme in this literature.[65]

One of the time-honored strategies for dealing with labor "problems" in the United States has been the use of female workers. There is a vast historical literature on this topic: from the very origins of the American industrial system women have been seen as a source of cheap, compliant, and undemanding labor.[66]

The same dynamic was a work in computer programming. In a 1963 *Datamation* article lauding the virtues of the female computer programmer, for example, Valerie Rockmael focused specifically on her stability, reliability, and relative docility: "Women are less aggressive and more content in one position …Women consider fringe benefits of more importance than their male peers and are more prone to stay on the job if they are content, regardless of a lack of advancement. They also maintain their original geographic roots and are less willing to travel or change job locations, particularly if they are married or engaged." In an era in which turnover rates for programmers *averaged* twenty percent annually, this was a compelling argument for employers. Note that this was something of a backhanded compliment, aimed more at the needs of employers than female programmers. In fact, the "most undesirable

category of programmers," Rockmael argued, was "the female about 21 years old and unmarried," because "when she would start thinking about her social commitments for the weekend, her work suffered proportionately."[67]

Women were often used in advertisements from this period as a visual proxy for low-skill, low-wage labor. For example, in its 1968 "Meet Susie Meyers" advertisements, the IBM Corporation suggested that even a "young girl" with "no previous programming experience" could program a computer using its new PL/1 programming language. The two-page, full-color advertisements showed a pretty blond in a colorful miniskirt dancing circles around her computer. If the problem with programming was that it was overly expensive ("Let's face it," the ad copy confided, "the cost of programming just keeps going up."), then the solution was the combination of mechanization and feminization. Although the advertisement promised "a brighter future for your programmers," the obvious subtext was that these programmers were becoming increasingly replaceable. If pretty little Susie Meyers, with her spunky miniskirt and utter lack of programming experience, could develop software effectively in PL/1, so could just about anyone.

These attempts to mobilize gendered rhetoric and visuals in the service of what one contemporary described as the "the domestication of this once proud, wild animal" (the computer programmer) did not go unnoticed by programmers.[68] "The Computer Girls" article, for example, prompted an almost immediate response from the Computer Sciences Corporation. Although the overlying tone of the advertisement was light-hearted—"In a recent issue of COSMOPOLITAN, Helen Gurley Brown exhorted her girl readers to become programmers and make 15,000 after five years …"—the underlying concern it expressed was also quite apparent: the suggestion that "Cosmo Girls" could make for good programmers was implicitly demeaning, and threatening to the status and future of the discipline.[69]

I have written extensively elsewhere about the "Question of Professionalism" as it emerged in the computer fields during the late 1960s.[70] For the purposes of this paper it is enough to note that the development of the structures of a programming profession—including formal programs in academic computer science, professional journals and societies, and professional certification programs—became the goal of many computer programmers, and their corporate employers, as a means of addressing the perceived "software crisis" of the late 1960s.

The professionalization of programming and other computer specialties was appealing to a number of constituencies. For practitioners, professionalism offered increased social status, greater autonomy, improved opportunities for advancement, and better pay. It provided individuals with a "monopoly of competence"—the control over a valuable skill that was readily transferable from organization to organization—

Susie Meyer meets PL/I

The story of how a single language answers the question, "Can a young girl with no previous programming experience find happiness handling both commercial and scientific applications, without resorting to an assembler language?"
Let's face it. The cost of programming just keeps going up. So for some time to come, how well you do your job depends on how programmers like Susie Meyer do theirs.

That's the reason for PL/I, the high-level language for both scientific and commercial applications.

With PL/I, programmers don't have to learn other high-level languages. They can concentrate more on the job, less on the language.

So think about PL/I. Not just in terms of training, but in terms of the total impact it can have on your operation.

Figure 5: IBM Corporation, Susie Meyers Meets PL/1, 1968

that provided leverage in the labor market.[71] Professionalism provided a means of excluding undesirables and competitors; it assured basic standards of quality and reliability; it provided a certain degree of protection from the fluctuations of the labor market; and it was seen by many workers as a means of advancement into the middle class.[72] The 1960s were a period when many white-collar occupations were pursuing professional agendas, and the sociological literature of period seemed to provide a clear road-map to the benefits of professionalism. These benefits seemed available to almost any occupation.[73]

The professionalization efforts of computer specialists were, to a certain extent, encouraged by their corporate employers. Professionalism provided a familiar solution to the increasingly complex problems of programmer management: "The concept of professionalism," argued one personnel research journal from the early 1970s, "affords a business-like answer to the existing and future computer skills market … The professional's rewards are full utilization of his talents, the continuing challenge and stimulus of new EDP situations, and an invaluable broadening of his experience base."[74] Insofar as it encouraged good corporate citizenship, professionalism had the potential to solve a number of pressing management problems: it might motivate staff members to improve their capabilities; it could bring about more commonality of approaches; it could be used for hiring, promotions and raises, and it could help solve the perennial question of "who is qualified."[75] At the very least, allowing programmers *think* that they were professionals would go a long way towards reducing turnover and maintaining the stability of the data processing staff.[76]

The desire to develop professional standards is an understandable, and indeed laudable, agenda for programmers to pursue. But it does carry with it certain implications for the gender dynamics of the discipline. As Margaret Rossiter and others have suggested, professionalization implies masculinization.[77] The imposition of formal educational requirements, such as a college degree, can make it difficult for women—particularly women who have taken time off to raise children—to enter the profession. Similarly, certification programs or licensing requirements—such as the Data Processing Management Association's Certificate in Data Processing Program—also erected barriers to entry that disproportionately affected women. In 1965, for example, the Association for Computing Machinery imposed a our-year degree requirement for membership which, in an era when he gender-ratio of male to female college undergraduates was close to 2-1, excluded significantly more women than men.[78] A survey from the late 1970s showed that fewer than 10% of ACM members were women.[79] Professionalism also suggests a certain degree of managerial authority and competence—skills and characteristics that were often seen as being masculine rather than feminine (see Hicks, this volume). The CDP examinations

explicitly required candidates to have at least three years of experience, and the majority of CDP holders worked in middle-management.[80] And in his 1971 book *The Psychology of Computer Programming*, Gerald Weinberg notes the commonly held belief that female programmers were incapable of leading a group or supervising their male colleagues.[81] The more programmers were seen as potential managers (a new development that came with professionalization), the more women were excluded.

There were other, more subtle ways in which professionalization implied masculinization. Perhaps most significantly, professionalization requires segmentation and stratification. In order to elevate the overall status of their discipline, aspiring professionals had to distance themselves from those aspects of their work that were seen as low-status and routine. This work did not just disappear—it was just done by other people. The job category of "programmer" had been used as a blanket term to describe a broad range of computer workers, but it was increasingly replaced by a complicated hierarchy of job titles: junior programmer, senior programmer, lead programmer, junior analyst, senior analysts, program manager, etc. Again, it is difficult to gather accurate statistics on who occupied what categories, but there is some evidence to suggest that women were generally confined to the lower levels of the professional pyramid.[82] This calls into question the more optimistic claims about the participation of women in computing: without knowing exactly what *kinds* of work these women were doing, it is difficult to draw any firm conclusions about the true nature of the opportunities available to women in computing.[83]

Conclusions

Contemporary discussions about the under representation of women in computing often center around the precipitous decline in female enrollments in academic computer science programs that started in the mid-1980s. But this sudden decline was only relative to an earlier and equally dramatic *increase* in female enrollments that occurred over the previous decades. In many ways it is this remarkable bulge in female enrollments that most deserves explanation. Compared to other scientific and engineering disciplines in this period, computer science—or at least computer programming, which was its closest analog prior to the institutionalization of the discipline in the late 1960s—was unusually welcoming to women. As "The Computer Girls" article in *Cosmopolitan* illustrates, and many other sources confirm, computing in its early years was seen as not only being unusually open to women, but also having unique advantages for women (the ability to work from home, for example).[84] It is only more recently that computer programming acquired its characteristically masculine identity. Unlike other technical or academic disciplines, which had been

traditionally male-dominated and had to be opened up to female participation, computer programming started out with an ambiguous gender identity. An activity originally intended to be performed by low-status, clerical—and more often than not, female—computer programming was gradually and deliberately transformed into a high-status, scientific, and masculine discipline.

The "masculinization" of computing was not universal or linear. Even as the computing fields were beginning to professionalize, women were continuing to work in computing in substantial numbers—as the continuing increase in computer science enrollments throughout the early 1980s indicates. To suggest that a discipline has been made masculine, however, is not to claim that all of its practitioners are male, but rather that the ideals of the discipline are masculine ideals. It is entirely possible, for example, to talk about science being gendered male without arguing that there are no female scientists.[85] To the degree that women succeed in masculinized disciplines, however, it is by suppressing their femininity: to act female in such contexts is to act "unprofessionally."[86] There is a large literature on the ways in which women in such fields are forced to accommodate themselves to the dominant gender dynamics of the discipline. The masculinization of a profession erects barriers to female participation, but it does not eliminate it altogether.

The history of the "computer girls" suggests at least two explanations for the remarkable occupational sex-change that occurred in computing over the course of the mid-20th century. The first is a structural argument, and suggests that masculinization is characteristic of any discipline that is actively professionalizing. In any case, seen through the lens of the history of the professionalization of the computing disciplines, the unusual pattern of female enrollments in computer science is slightly more explicable. In the early decades of computing, before the discipline was effectively professionalized, the field was much more open to female participation. The additional opportunities promised by the emergence of the personal computer might explain the final surge of the late 1970s. But eventually the development of the structures of a profession—a slow but the steady process that had started decades earlier—brought to an end the era of unprecedented openness in computing, and brought enrollments in computer science programs back in line with other scientific, mathematical, and engineering disciplines. In this sense, while enrollments in computer science programs are an extremely inadequate measure of female participation in computing overall, it is a reasonable measure of the professionalization and masculinization of the discipline. In fact, if we interpret the formation of academic computer science programs as a crucial contributor to the masculinization of programming, rather than a measure of its degree, then the focus of the conversation changes fundamentally. Instead of asking why so few women in computer science, we

might ask instead why a particular vision of the discipline—one based on masculine ideals and values—came to dominate the academic study of computer programming.

This structural explanation is not entirely sufficient, however. Although patterns in computer science enrollments do resemble those of other scientific disciplines (and perhaps even more those of engineering programs), it also has its own, distinctively masculine culture. Many observers have identified this culture as being particularly unappealing to women. The popular association of computing culture with the "nerd" stereotype is perhaps the most common explanations for low rates of participation among females. In recent decades the "computer nerd" has become a staple of modern American culture, and is invariably represented as eccentric, unkempt, anti-social—and male.

The story of the computer "nerd" is often associated with the personal computer. A powerful mythology has developed around the role of the nerdy loner in the "accidental" creation of the personal computer industry.[87] The presence of white, adolescent, male nerds is often represented as the essential characteristic of any successful technological start-up company. Nerd culture supposedly dominates most modern computer science departments.

As we have seen, however, the social construction of the computer programmer as a nerdy eccentric predates the personal computer by several decades. It originated in the early association of programming ability with chess-playing and mathematics puzzles, was reinforced by scientifically dubious aptitude tests and personality profiles, and by the early 1960s had become embodied in the hiring practices of the growing commercial computer industry. The institutionalization of gender norms in this period highlights the ways in which structure and culture are mutually constitutive, and ultimately self-replicating. Even as underlying structural explanations disappear, the cultural superstructure remains intact.

One simple but powerful example of this relationship has to do with the development of the "nocturnal" culture of computing. In an era when computers were large, expensive machines that ran in batch-production mode, computer programmers often had unfettered access to the computer only during off-hours, which often meant over-night. In some cases, this represented a tangible structural barrier to female participation: some corporations specifically prohibited women from remaining on-premises after business hours (ostensibly for safety reasons), which effectively prevented these women from working as programmers.[88] But even after the technical requirements for such nocturnal programming activities disappeared, the culture of staying up all night and ignoring the normal conventions of 24-hour time continued to persist, and, in fact, be celebrated, within certain computing communities.[89] To the degree to which these practices are unappealing or impractical for women reflects

the close interaction between culture and structure in the replication of gender norms and identity. What seem to contemporaries like the "natural" way in which things have "always" been done are historically contingent.

It is this relationship between structure and culture that reveal most clearly the value of the history of computing to the contemporary practice of computing. Ideas about *how* computing should be done corresponded closely with perceptions of *who* should be doing the computing. In the case of computer programming, these ideas and perceptions changed dramatically over the course of the mid-20th century, often in ways in which were invisible to practitioners. The widespread adoption of aptitude testing by corporate employers, for example, was not deliberately aimed at excluding women, and in fact might in other circumstances served to expand opportunities for female participation. But the particular ways in which aptitude tests and personality profiles were developed, and the ways in which these tests and profiles were used in the context of other efforts to define what computer programming was and who should being doing it, had unintended consequences. These consequences became embodied in the structures of the industry. The gender identity and culture of computing became fixed, and ultimately self-perpetuating, as these structures became normalized.

Notes

[1] Lois Mandel. "The Computer Girls," *Cosmopolitan* (1967), pp. 52-56.

[2] Ibid.

[3] Richard Canning. "Issues in Programming Management," *EDP Analyzer* 12 no. 4 (1974), pp. 1-14.

[4] Bruce Gilchrist and Richard Weber. "Enumerating Full-Time Programmers," *Communications of the ACM* 17 no. 10 (1974), pp. 592-593.

[5] Mandel, "Computer Girls"

[6] A.M. Koss. "Programming on the Univac 1: a woman's account," *IEEE Annals of the History of Computing* 25 no. 1 (2003), pp. 48-59; S. M. Campbell. "Beatrice Helen Worsley: Canada's Female Computer Pioneer," *IEEE Annals of the History of Computing* 25 (2003).

[7] Mandel, "Computer Girls"

[8] Robert Patrick. "The Gap in Programming Support," *Datamation* 7 no. 5 (1961), p. 37; Don Madden. "The Population Problem: Inexperience Will Dominate," *Datamation* 8 no. 1 (1962), p. 26.

[9] "Careers in Computers" [advertisement], *Datamation* 8 no. 1 (1962), p. 80, p. 21.

[10] "Software Gap – A Growing Crisis for Computers," *Business Week* (Nov. 1966).

[11] "Not Quite All About MIS," *Datamation* 13 no. 5 (1967), p. 21.

[12] Edward Markham. "EDP Schools - An Inside View," *Datamation* 14 no. 4 (1968), pp. 22-27.

[13] Richard Tanaka. "Fee or Free Software," *Datamation* 13 no. 10 (1967), pp. 205-206.

[14] Gene Bylinsky. "Help Wanted: 50,000 Programmers," *Fortune* 75 no. 3 (1967), pp. 445-556, p. 141.

[15] Tanaka. "Fee or Free Software," pp. 205-206.

[16] Jean P. Gilbert and David B. Mayer. "Experiences in self-selection of disadvantaged people into a computer operator training program," *SIGCPR '69: Proceedings of the seventh annual conference on SIGCPR*. New York, NY, USA: ACM Press, 1969, pp. 79-90.

[17] "First Programmer Class at Sing Sing Graduates," *Datamation* 14 no. 6 (1968), pp. 97-98.

[18] Bylinsky, "Help Wanted: 50,000 Programmers."

[19] Ibid., p. 141.

[20] Mandel, "Computer Girls", p. 56.

[21] Charles Lawson. "A Survey of Computer Facility Management," *Datamation* 8 no. 7 (1962), pp. 29-32.

[22] W. J. McNamara. "The selection of computer personnel: past, present, future," *SIGCPR '67: Proceedings of the fifth SIGCPR conference on Computer personnel research*. New York, NY, USA: ACM Press, 1967, pp. 52-56.

[23] Jennifer Light. "When Computers Were Women," *Technology & Culture* 40 no. 3 (1999), pp. 455-483.

[24] A. Goyal. "Women in computing: historical roles, the perpetual glass ceiling, and current opportunities," *IEEE Annals of the History of Computing* 18 no. 3 (1996), pp. 36-42.

[25] Judy Wajcman. "Reflections on Gender and Technology: In What State is the Art?" *Social Studies of Science* 30 no. 3 (2000), pp. 447-464.

[26] Janet Abbate. "How did you first get into computing?" *IEEE Annals of the History of Computing* 25 (2003), pp. 4-8.

[27] W. Barkley Fritz. "The Women of Eniac," *Annals of the History of Computing* 18 no. 3 (1996), pp. 13-23, p. 20.

Figure 6: Datamation Magazine Advertisement, 1966

Figure 7: Datamation Magazine Advertisement, 1967

[28] Light, "When Computers Were Women."

[29] John Tukey. "The Teaching of Concrete Mathematics," *American Mathematical Monthly* 65 no. 1 (1958), pp. 1-9.

[30] David Allan Grier. "The ENIAC, the verb to program, and the Emergence of Digital Computers," *Annals of the History of Computing* 18 no. 1 (1996), pp. 51-55, p. 53.

[31] Herman Goldstine and John von Neumann, "Planning and Coding of Problems for an Electronic Computing Instrument" (Institute for Advanced Study, Princeton, NJ, 1947)

[32] "Introduction to Programming: Programming for the Univac, Part 1," Typewritten manuscript, dated 11 June 1949. Hagley Archives, Box 372, Accession 1825.

[33] Grier, "The ENIAC,"Error: Reference source not found p. 52.

[34] Richard Wexelblat, ed. *History of Programming Languages*. (New York: Academic Press, 1981), p. 69.

[35] Margery Davies. *Woman's Place is at the Typewriter: Office Work and Office Workers, 1870-1930*. (Philadelphia: Temple University Press, 1982); Sharon Hartman Strom. *Beyond the Typewriter: Gender, Class, and the Origins of Modern American Office Work, 1900-1930*. (Urbana: University of Illinois Press, 1992); Elyce J Rotella. *From Home to Office: U.S. Women at Work, 1870-1930*. (Ann Arbor, Mich.: UMI Research Press, 1981).

[36] Thomas Haigh. "The Chromium-Plated Tabulator: Institutionalizing an Electronic Revolution, 1954-1958," *IEEE Annals of the History of Computing* 4 no. 23 (2001), pp. 75-104.

[37] Jackson Granholm. "How to Hire a Programmer," *Datamation* 8 no. 8 (1962), pp. 31-32; H.A. Rhee. *Office Automation in Social Perspective: The Progress and Social Implications of Electronic Data Processing*. (Oxford: Basil Blackwell, 1968).

[38] Edsger Dijkstra. "The Humble Programmer," *Communications of the ACM* 15 no. 10 (1972), pp. 859-866.

[39] Edsger Dijkstra. "Programming as a Discipline of Mathematical Nature," *American Mathematical Monthly* 81 no. 6 (1974), pp. 608-612.

[40] Martin Campbell-Kelly and William Aspray. *Computer: A History of the Information Machine*. (New York: Basic Books, 1996), p. 201.

[41] Joseph O'Shields. "Selection of EDP Personnel," *Personnel Journal* 44 no. 9 (1965), pp. 472-474.

[42] Nathan Ensmenger, "Chess-players, Music-lovers, and Mathematicians ," in *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise* (MIT Press, forthcoming).

[43] Mark Halpern. "Memoirs (Part 1)," *Annals of the History of Computing* 13 no. 1 (1991), pp. 101-111.

[44] "The Computer Personnel Research Group," *Datamation* 9 no. 1 (1963), pp. 38-39.

[45] Hal Sackman, W.J. Erickson, and E.E. Grant. "Exploratory Experimental Studies Comparing Online and Offline Programming Performance," *Communications of the ACM* 11 no. 1 (1968), pp. 3-11.

[46] P. Mody. "Is Programming an Art?" *Software Engineering Notes* 17 no. 4 (1992), pp. 19-21; Maurice Black, *The Art of Code* (PhD Dissertation, University of Pennsylvania, 2002)

[47] Rhee, *Office Automation in Social PerspectiveError: Reference source not found*.

[48] Except to the extent that fraternities and other male social organizations served as clearinghouses for stolen copies of popular aptitude tests such as the IBM PAT. Such theft and other forms of cheating were rampant in the industry, and taking the test more than once was almost certain to lead to a passing grade.

[49] William Paschell. *Automation and Employment Opportunities for Office Workers; a Report on the Effect of Electronic Computers on Employment of Clerical Workers*. (Washington, DC: Bureau of Labor Statistics, 1958); Gerald Weinberg, *The Psychology of Computer Programming*. (New York: Van Nostrand Rheinhold, 1971).

[50] Dallis Perry and William Cannon. "Vocational Interests of Computer Programmers," *Journal of Applied Psychology* 51 no. 1 (1967), pp. 28-34.

[51] Ibid.

[52] Ibid.

[53] Dallis Perry and William Cannon. "Vocational Interests of Female Computer Programmers," *Journal of Applied Psychology* 52 no. 1 (1968), p. 31.

[54] Richard Brandon, "*The problem in perspective*," *Proceedings of the 1968 23rd ACM national conference* (ACM Press, 1968), pp. 332-334.

[55] *Needs, interests, and reinforcer preferences of data processing personnel*. Vol. Proceedings of the eighth annual SIGCPR conference. 1970.

[56] Weinberg, *The Psychology of Computer Programming*

[57] William Ledbetter. "Programming Aptitude: How Significant is It?" *Personnel Journal* 54 no. 3 (1975), pp. 165-166, 175.

[58] Jane Margolis and Allan Fisher. *Unlocking the Clubhouse: Women in Computing*. (Cambridge, Mass.: MIT Press, 2002); Jennifer Taylor. "The Decline of Women in Computer Science, 1940-1982" (MA Thesis, Harvard University Graduate School of Education, 2005); J. McGrath Cohoon and William Aspray. *Women and Information Technology: Research on Underrepresentation*. Cambridge, Mass.: MIT Press, 2006.

[59] Hal Sackman. "Conference on Personnel Research," *Datamation* 14 no. 7 (1968), pp. 74-76, 81.

[60] The most damning critique of the "black art" of programming came from Douglas McIroy at the 1968 NATO Conference on Software Engineering: "We undoubtedly produce software by backward techniques. We undoubtedly get the short end of the stick in confrontations with hardware people because they are the industrialists and we are the crofters. Software production today appears in the scale of industrialization somewhere below the more backward construction agencies. I think that its proper place is considerable higher, and would like to investigate the prospects for mass-production techniques in software."

[61] John Golda. "The Effects of Computer Technology on the Traditional Role of Management," MA thesis. Wharton School of Business, University of Pennsylvania, 1965.

[62] Joan Greenbaum. "On twenty-five years with Braverman's 'Labor and Monopoly Capital'. (Or, how did control and coordination of labor get into the software so quickly?)" *Monthly Review* 50 no. 8 (1999): 28-42.

[63] Greg Downey, "Commentary: The Place of Labor in the History of Information-Technology Revolutions," *International Review of Social History* 48 no. 11 (2003), pp. 225-261.

[64] Nathan Ensmenger. "From 'Black Art' to Industrial Disciple: The Software Crisis and the Management of Programmers," PhD thesis. University of Pennsylvania, 2001.

[65] Michael Cusamano. "Factory Concepts and Practices in Software Development," *Annals of the History of Computing* 13 no. 1 (1991), pp. 3-32.

[66] Ruth Milkman, *Gender at work: The dynamics of job discrimination by sex during World War II* (Urbana: University of Illinois Press, 1987); Alice Kessler-Harris, *Out to work: a history of wage-earning women in the United States* (New York, Oxford University Press, 1982).

[67] Valerie Rockmael. "The Woman Programmer," *Datamation* 9 no. 1 (1963), p. 41.

[68] Datamation Editorial. "Of Maturity and Meatballs," *Datamation* 9 no. 8 (1963), p. 23.

[69] Computer Sciences Corporation. "In case you missed our first test..." *Datamation* 13 no. 9 (1967), p. 149.

[70] Nathan Ensmenger. "The 'Question of Professionalism' in the Computer Fields," *IEEE Annals of the History of Computing* 23 no. 4 (2001), pp. 56-73; Nathan Ensmenger and William Aspray, "Software as a Labor Process," *History of Computing: Software Issues*. Ed. by Ulf Hashagen, Reinhard Keil-Slawik, and Arthur Norberg. (Berlin/New York: Springer-Verlag, 2002).

[71] Magali Sarfatti Larson. *The Rise of Professionalism: A Sociological Analysis*. (Berkeley: University of California Press, 1977).

[72] Robert Zussman. *Mechanics of the middle class: Work and politics among American engineers*. (Berkeley: University of California Press, 1985).

[73] The sociologist Harold Wilensky describes numerous case studies of occupations attempting to professionalize in this period, among them librarians, druggists, funeral directors, and high-school teachers. See Harold Wilensky, "The Professionalization of Everyone?" *American Journal of Sociology* 70.2 (1964), pp. 137–158.

[74] "Professionalism Termed Key to Computer Personnel Situation," *Personnel Journal* 51 no. 2 (1971), pp. 156-157.

[75] Richard Canning. "Professionalism: Coming or Not?" *EDP Analyzer* 14 no. 3 (1976), pp. 1-12.

[76] Robert Gordon. "Personnel Selection," in Fred Gruenberger and Stanley Naftaly, eds. *Data Processing...Practically Speaking* (Los Angeles, Data Processing Digest, 1967), pp. 87-88.

[77] Margaret Rossiter. *Women Scientists in America: Struggles and Strategies to 1940*. (Baltimore: Johns Hopkins University Press, 1982); Jeffrey Hearn. "Notes on Patriarchy, Professionalization and the Semi-Professions," *Sociology* 16 .2 (1982), pp. 184-202; Ruth Oldenziel. *Making Technology Masculine: Men, Women and Modern Machines in America, 1870-1945*. (Amsterdam: Amsterdam University Press, 1999).

[78] Claudia Goldin, Lawrence Katz, and Ilyana Kuziemko. "The Homecoming of American College Women: The Reversal of the College Gender Gap," *Journal of Economic Perspectives* 20 no. 4 (2006), pp. 133-156.

[79] Thomas D'Auria. "ACM Membership Profile Report," *Communications of the ACM* 20 no. 10 (1977), pp. 688-692.

[80] Theodore Willoughby. "Psychometric Characteristics of the CDP Examination." Proceedings of the Thirteenth Annual SIGCPR conference (ACM Press, 1975), pp. 152-160.

[81] Weinberg, *The Psychology of Computer ProgrammingError: Reference source not found*, p. 85.

[82] Richard Weber and Bruce Gilchrist. "Discrimination in the Employment of Women in the Computer Industry," *Communications of the ACM* 18 (1975), pp. 416-418.

[83] Beverly H. Burris. "Technocracy and Gender in the Workplace," *Social Problems* 36 no. 2 (1989), pp. 165-180.

[84] A. M. Koss. "Programming at Burroughs and Philco in the 1950s," *IEEE Annals of the History of Computing* 25 no. 4 (2003): 40-50.

[85] Rossiter, *Women Scientists in America*; Evelyn Fox Keller, "Gender and Science: Origin, History, and Politics," *Osiris* 10 (1995), pp. 27-38.

[86] Carol Cohn, "War, Wimps and Women: Talking Gender and Thinking War," in M. Cooke and A. Woolcott, eds., *Gendering War Talk* (Princeton, Princeton University Press Princeton, 1993), pp. 227-246.

[87] Robert Cringely, *Accidental Empires: How the Boys of Silicon Valley Make Their Millions, Battle Foreign Competition and Still Can't Get a Date* (Penguin Books, 1996).

[88] Weinberg, op. cit.Error: Reference source not found

[89] Joseph Weizenbaum, *Computer Power and Human Reason: From Judgment to Calculation*. MIT Press, 1976; Steven Levy, *Hackers: Heroes of the Computer Revolution*. (Garden City NY: Anchor Press/Doubleday, 1984); Sherry Turkle. *The Second Self: Computers and the Human Spirit*. (New York: Simon and Schuster, 1984).