The 'Question of Professionalism' in the Computer Fields

Nathan L. Ensmenger
University of Pennsylvania

In the late 1950s and early 1960s, the "question of professionalism" became a pressing issue for the emerging commercial computer industry. Just who was qualified to be a programmer? Competing visions as to the answers to these questions contributed to an ongoing debate that caused turf wars, labor shortages, and varied approaches to professional development. The author explores the many diverse attitudes and opinions on what professionalism meant in the 1950s and 1960s.

> In one inquiry 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. The last was considered the least competent.[1]
>
> H.A. Rhee,
> *Office Automation in Social Perspective* (1968)

The first computer programmers came from a variety of occupational and educational backgrounds. Some were former clerical workers or tabulating machine operators. Others were recruited from the ranks of the female "human computers" who had participated in wartime manual computation projects. Most, however, were erstwhile engineers and scientists recruited from military and scientific hardware development projects. For these well-educated computer "converts," it was not always clear where computer programming stood in relation to more traditional disciplines. In the early 1950s, the disciplines that we know today as computer science and software engineering existed only as a loose association of institutions, individuals, and techniques. Although computers were increasingly used in this period as instruments of scientific production, their status as legitimate objects of scientific and professional scrutiny had not yet been established. Those scientists who left "respectable" disciplines for the uncharted waters of computing faced self-doubt, professional uncertainty, and even ridicule. The physicist-turned-computer-scientist Edsgar Dijkstra recalled this difficult process of self-transformation in his 1972 Turing Award Lecture (revealingly titled "The Humble Programmer"):

> I had to make up my mind, either to stop programming and become a real, respectable theoretical physicist, or to carry my study of physics to formal completion only, with a minimum of effort, and to become … what? A programmer? But was that a respectable profession? After all what was programming? Where was the sound body of knowledge that could support it as an intellectually respectable discipline? I remember quite vividly how I envied my hardware colleagues, who, when asked about their professional competence, could at least point out that they knew everything about vacuum tubes, amplifiers and the rest, whereas I felt that, when faced with that question, I would stand empty-handed.[2]

Dijkstra was by no means alone in his assessment of the ambiguous professional status of computing personnel. Over the previous decade, computing had managed to acquire many trappings of a profession: research laboratories and institutes, professional conferences, professional societies, and technical journals. Indeed, as William Aspray has suggested, many of the structural elements of a computing profession were in place by the end of the 1950s.[3] But the existence of professional institutions did not necessarily translate readily into widely recognized professional status. Throughout the 1960s, computer specialists continued to wonder at the "almost universal contempt" (or at least "cautious bewilderment and misinterpretation") with which programmers were regarded by the general public.[4] In 1967, the US Civil Service Commission declared data-processing personnel to be nonexempt employees, officially categorizing programmers and other computer specialists as technicians rather than professionals. In a series of articles discussing the "question of professionalism," the influential computer industry analyst Richard Canning declared that "true professional status for systems analysts and/or programmers" was "no closer today [in 1975] than it was ten years ago. It is not clear just what body of practitioners should rightly classify as professionals."[5]

The question of who was qualified to be a computer professional was of interest to more than just practitioners. In their role as mediators between the computer and the larger society, computer professionals have always played a crucial role in defining what the computer was and what it could be used for. As the electronic computer became an increasingly valuable source of institutional and economic power and authority, programmers and other computer personnel emerged as influential organizational "change-agents" (to use the management terminology of the era).[6] This was particularly true of business programmers. The systems they developed often replaced, or at least substantially altered, the work of traditional white-collar employees. Traditional corporate managers, not unsurprisingly, often resented the perceived impositions of the "computer boys," regarding them as threats to their position and status.[7] They attempted to reassert control over operational decision making by defining programmers as narrow specialists or "mere technicians."[8] The result was a highly contested struggle over the proper place of the programmer in traditional academic, occupational, and professional hierarchies.

This article explores the "question of professionalism" as it applied to the computing disciplines in the 1950s and 1960s. This was a period of particularly intense professional activity by computer specialists, particularly those involved with software development. The rapid expansion of the commercial computer industry, the rising cost of software relative to hardware, and the widespread perception that a software crisis was imminent lent urgency to the debate about the character and future of the computing professions. Would programmers and other computer specialists survive as skilled, autonomous professionals, or would they be supplanted by "automatic programming" languages or mass-produced software components? Was computer science a legitimate, independent discipline, or simply a "momentary aberration in the fields of mathematics and electrical engineering"?[9] Should the profession model itself along the lines of the research scientist, industrial engineer, or certified public accountant? Were the ideal computer professionals college-educated or merely vocational school graduates? Were they professionally certified or licensed by the state? The debate about the nature and character of computer professionals that took shape in this period was part of an ongoing debate about information technology workers that spans the decades from the 1950s to the present. Many of this debate's most persistent themes—the contested relationship between computer science curricula and the skills required of business programmers, for example—cannot be fully understood outside the context of this defining period in the history of the computing professions.

The persistent personnel problem

The debate about professionalism in the computer field originated in the programmer labor shortages of the early 1950s. The supply of programmers had been a problem for the commercial computing industry from the beginning. As early as 1954, the organizers of the first Conference on Training Personnel for the Computing Machine Field warned that "estimates of manpower needs for computer applications … [are] astounding compared to the facilities for training people for this work."[10] With the annual demand for programmers expected to double, there was among participants "a universal feeling that there is a definite shortage of technically trained people in the computer ."[11] The largest employer (and trainer) of programmers in this period, the System Development Corporation (SDC), could hardly train enough programmers to meet its own internal demand.[12]

As the market for commercial computers changed and expanded in the early 1960s, the demand for computer specialists increased accordingly. The rapidly growing "gap in programming support" threatened to explode into a full-blown crisis, and it appeared that the situation would "get worse in the next several years before it gets better."[13] In 1962, the editors of the data-processing trade 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 in a variety of flavors."[14] Five years later, "one of the prime areas of concern" to electronic data-processing (EDP) managers was still "the shortage of capable programmers," a shortage that had "profound implications, not only for the computer industry as it is now, but for how it can be in the future."[15] In 1967, an influential report on "The State of the Information Processing Industry" 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.[16] "Competition for programmers has driven salaries up so fast," warned a contemporary article in *Fortune* magazine, "that programming has become probably the country's highest paid technological occupation …. Even so, some companies can't find experienced programmers at any price."[17]

Faced with a growing shortage of skilled programmers, employers were forced to pursue increasingly desperate measures to staff their software development projects. Many were forced to develop expensive internal training programs, "not because they want to do it, but because they have found it to be an absolute necessary adjunct to the operation of their business."[18] In 1966, IBM provided programming training for 100,000 people at a cost of $90 to $100 million.[19] New sources of manpower production (some legitimate, others less so) emerged to meet the demand for trained programmers. Private data-processing schools sprang up all over the country promising high salaries and dazzling career opportunities.[20] A 1968 article in *Cosmopolitan* magazine urged Helen Gurley Brown's "Cosmo Girls" to go out and become "computer girls" making "$15,000 a year" as programmers.[21] At one point, the so-called "population problem" in software became so severe that service bureaus in New York farmed out programming work to inmates at the nearby Sing-Sing prison, promising them permanent positions pending their release.[22]

The influx of new programmer trainees and vocational school graduates into the software labor market failed to alleviate the acute shortage of programmers plaguing the industry, however. In fact, one 1968 study by the Association for Computing Machinery's Special Interest Group on Computer Personnel Research (SIGCPR) warned of 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. Educational, performance, and professional standards are virtually nonexistent and confusion grows rampant in selecting, training, and assigning people to do jobs.[23]

The openness of the computing field, which was seen by many early entrants as its most attractive quality—"In what other field are you likely to find a Ph.D. and a person whose education stopped at the high school level working as equals on the same difficult technical problem?"[24]—was disconcerting to conventional personnel directors. "It is not unusual to find college graduates as well as high school drop-outs programming computers," warned the editors of the *Personnel Journal*, "and making technical decisions within their programs, which may affect company operations."[25] It quickly became apparent that the labor shortage in computing was not so much a lack of computer specialists per se; what the industry was really worried about was a shortage of experienced, professional practitioners. That there was little agreement within the computing community about who exactly qualified as an experienced, professional practitioner only served to emphasize their real or perceived rarity.

The lack of established standards for who qualified deterred many of the aspiring programmer trainees who might otherwise have helped alleviate the growing labor shortage. The lack of a clear point of entry into the discipline discouraged many potential candidates. "I have heard about this 'extreme shortage of programmers,'" wrote one *Datamation* reader, but "no one wants trainees. How does a person … get into programming?"[26] Pleaded another, "Could you answer for me the question as to what in the eyes of industry constitutes a 'qualified' programmer? What education, experience, etc. are considered to satisfy the 'qualified' status?"[27] The frequent scandals that plagued the private vocational training programs prompted some companies to adopt "no data-processing school graduate" policies, effectively excluding large pools of would-be employees.[28] Employers and programmers alike were anxious to produce better standards for training and curriculum, but it was unclear to whom they should turn for guidance.

The obvious candidates for establishing standards for programming competency were the universities. Although computer science in the late 1950s and early 1960s was not yet an established discipline, many of the larger research universities were beginning to offer graduate training in computer-related specialties. Because academic computer scientists were struggling in this period to define a unique intellectual identity for their discipline, they focused on developing a theoretical basis for their discipline, rather than providing training in practical techniques.

As computing became more business oriented, the mismatch between university training and the needs of employers became increasingly apparent. Many corporations saw these university programs—most of which focused on formal logic and numerical analysis—as being increasingly out-of-touch with the needs of their business. As the computer scientist Richard Hamming pointed out in his 1968 Turing Award Lecture:

Their experience is that graduates in our programs seem to be mainly interested in playing games, making fancy programs that really do not work, writing trick programs, etc., and are unable to discipline their own efforts so that what they say they will do gets done on time and in practical form.[29]

If the discipline were going to turn out "responsible, effective people who meet the real needs of our society," Hamming suggested, computer science departments must abandon their love affair with pure mathematics and embrace a hands-on engineering approach to computer science education.

Hamming was hardly the only member of the computing community to find fault with the increasingly theoretical focus of contemporary computer science. As early as 1958, a US Bureau of Labor report on *The Effects of Electronic Computers on the Employment of Clerical Workers* had noted a growing sense of corporate disillusionment with academic computer science:

Many employers no longer stress a strong background in mathematics for programming of business or other mass data if candidates can demonstrate an aptitude for the work. Companies have been filling most positions in this new occupation by selecting employees familiar with the subject matter and giving them training in programming work.[30]

Academic computer scientists sought to reinvent the programmer in the model of the research scientist; corporate employers resisted what they saw as "a sort of holier than thou academic intellectual sort of enterprise."[31] The tension between these competing visions of what a programmer should be only served to further exacerbate the shortage of qualified programmers.[32]

In any case, the relatively small number of colleges and universities that did offer some form of practical computer experience were unable to provide trained programmers in anywhere near the quantities required by industry. As a result, aspiring software personnel often pursued alternative forms of vocational training. Some were recruited for in-house instruction programs provided by employers or computer manufacturers. Others enrolled in the numerous private EDP training schools that began to appear in the mid-1960s. Unfortunately, many of these schools were profit-oriented enterprises more interested in quantity than quality, whose "only meaningful entrance requirements are a high school diploma, 18 years of age … and the ability to pay."[33] The more legitimate programs suffered from many of the same problems that plagued the universities: a shortage of experienced instructors, the lack of established standards and curricula, and general uncertainty about what skills and aptitudes made for a qualified programmer. The problem was not only that the universities and vocational schools could not provide the type of educational experience that interested corporate employers; the real issue was that most employers were simply not at all sure what they were looking for.

The drive toward professionalism

Employers were not the only ones concerned with the labor problem in programming. Experienced computer personnel were keenly aware of the labor crisis and the tension it was producing in their industry and profession, as well as in their individual careers. Although computer specialists in general were appreciative of the short-term benefits of the labor shortage (above-average salaries and plentiful opportunities for occupational mobility), many believed that a continued crisis threatened the long-term stability and reputation of their industry and profession. As a *Datamation* editorial put it,

With a mounting tide of inexperienced programmers, new-born consultants, and the untutored outer circle of controllers and accountants all assuming greater technical responsibility, a need for qualification of competence is clearly apparent.[34]

The software community's inability to provide its own solution to the certification problem within EDP, warned some observers, "will result in a solution imposed from without. In several fields, the lack of professional and industrial standards has prompted the government to establish standards."[35]

Computer programmers in particular were concerned that an influx of the kind of "narrow, semi-literate technicians" being ground out by trade schools and junior colleges would damage the reputation of their profession.[36] "As long as anyone with ten dollars can join the ACM and proclaim himself a professional computer expert," it would be impossible to "guarantee the public a minimum level of competence in anyone who is permitted to claim membership in the profession."[37] Some industry leaders felt that the "low status" of computer programming would deter promising candidates from entering the discipline.[38] Others worried about incursions by other, more established professions into what software workers regarded as their own proprietary, occupational territory:

> We can wait for the CPA types to find out the tricks of our trade, train a substantial number of their younger sub-alterns in machines and programming languages, and take over the task. Or we can establish a parallel license, team up with the CPA's for accounting and auditing tasks, and work in other directions independently.[39]

In the sociological literature of the era, jurisdictional control over training and certification was presented as the sine qua non of professionalism. Without the ability to decide "who was qualified," computer specialists had no right to consider themselves true professionals.

Concerns about the long-term future of their occupation also weighed heavily on the minds of many computer specialists, particularly programmers. What was the appropriate career path for a software worker? "There is a tendency," suggested the ACM SIGCPR,

> for programming to be a 'dead-end' profession for many individuals, who, no matter how good they are as programmers, will never make the transition into a supervisory slot. And, in too many instances this is the only road to advancement.[40]

Whereas traditional engineers were often able (and, in fact, expected) to climb the corporate ladder into management positions, programmers were often denied this opportunity.[41] It was not clear to many corporate employers how the skills possessed by programmers would map onto the skills required for management. Although some companies offered alternative career ladders for programmers, "in actual practice, technical careers [were] not comparable to managerial careers, in pay, prestige or responsibility."[42]

Many of the job advertisements from this period reflected the concerns that programmers had regarding their occupational future and longevity. SDC, for example, emphasized "the large number of supervisory positions open to programmers [at SDC] and the fact that most programming supervisors have programming backgrounds."[43] Other companies offered similar appeals to the long-term career aspirations of programmers:

> At Xerox, we look at programmers … and see managers.[44]

> Working your way towards obsolescence? At MITRE professional growth is limited only by your ability.[45]

> Is your programming career in a closed loop? Create a loop exit for yourself at [the Bendix Corporation].[46]

Although starting salaries were high and individual programmers were able to move with relative ease horizontally throughout the industry, there were precious few opportunities for vertical advancement.[47] Many programmers worried about becoming obsolete and felt pressure to constantly upgrade their technical skills.[48] The result was a curious tension between a general sense of optimism and unlimited opportunity on the one hand, and fear and uncertainty about long-term career prospects on the other.

Programmers and other computer specialists were also concerned about what many saw as disturbing new trends in corporate attitudes toward computer personnel. By the end of the 1960s, the debate over programmer training and recruitment had been elevated to the level of a national crisis. Faced with rising software costs, and threatened by the unprecedented degree of autonomy that high-level executives seemed to grant to "computer people," many corporate managers began to reevaluate their largely hands-off policies toward programmer management. Whereas in the 1950s computer programming was widely considered to be a uniquely creative activity—and therefore almost impossible to manage using conventional methods—by the end of the 1960s, new perspectives on these problems began to appear in the industry literature.[49] The same qualities that had previously been thought essential indicators of programming ability, such as creativity and a mild degree of personal eccentricity, now began to be perceived as being merely unprofessional. Traditional managers began to accuse programmers and other computer personnel of lacking professional standards and loyalties: "too frequently these people, while exhibiting excellent technical skills, are non-professional in every other aspect of their work."[50]

In an attempt to free themselves from what an increasing number saw as a dangerous dependency on programmer labor, managers attempted to develop alternative solutions to the "persistent personnel problem." Many of these solutions were technical in nature, such as the so-called automatic-programming systems that promised to enable users to program their computers directly, thereby "eliminating the middleman."[51] Others represented attempts to "rationalize" the process of software development by applying to it the lessons learned from traditional manufacturing: replaceable parts, simple and repetitive tasks, and a strict division of labor. In one widely quoted paper on "mass-produced software components," for example, Douglas McIlroy articulated his plan for industrializing software production:

> 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 considerably higher, and would like to investigate the prospects for mass-production techniques in software.[52]

Although McIlroy does not explicitly address issues of professional concern to computer specialists—such as status, autonomy, and job satisfaction—his vision of a software "components factory" invoked familiar images of industrialization and proletariatization. Programmers in these factories need only be trained to perform a limited and specialized function, and could effectively be looked upon as interchangeable units.[53] They would be encouraged to be professionals only to the extent that being a professional meant self-discipline, a willingness to work long hours with no overtime pay, and loyalty to the corporation and obedience to supervisors.[54] As the noted computer scientist Andrei Ershov warned in his 1972 keynote address to the Spring Joint Computer Conference, programmers were in danger of losing their professional identity and becoming "what is simply a highly paid subgroup of the working class."[55] He continued,

> Even the claim of programmers to be a special breed of professional employee has come to be disputed. Still more significant, authority over the freewheeling brotherhood of programmers is slipping into the paws of administrators and managers who try to make the work of programmers planned, measurable, uniform, and faceless.[56]

Although it is possible to overemphasize the degree to which computer personnel in this period were in danger of becoming an "oppressed and degraded workforce," clearly the fear of having their occupation "routinized" motivated many computer specialists to give thought to their professional identity.[57]

Faced with these unsettling challenges to their professional identity, many computer specialists began to take renewed interest in issues of professional development. In the mid-1960s, articles on various aspects of professionalism—the establishment of a standard curriculum for computer science education, support for industry-based certification and licensing programs, and the introduction of professional codes of ethics—appeared with increasing frequency in industry journals such as the *Communications of the ACM* and *Datamation*. Transcripts from the annual Rand-sponsored computing symposia from this period indicate that professional concerns weighed heavily on the minds of many of the most influential leaders of the computing community.[58] Explicit comparisons were made to established professions such as law and medicine.[59] In 1962, the Data Processing Management Association (DPMA) announced its ambitious "Six Measures of Professionalism Program," which included provisions for certification standards, continuing education, public service, and the development of a professional code of ethics.[60] In 1966, the ACM announced a $45,000 professional development program that included "skill upgrade" seminars offered at the national computer conferences, a traveling course series, and self-study materials.[61] Both efforts appear to have been responses to a larger groundswell of support for increased professionalism in the computer fields.

The professionalization of programming and other computer specialties was appealing to practitioners because professionalism offered increased social status, greater autonomy, improved opportunities for advancement, and better pay.[62] It provided individuals with a "monopoly of competence"—the control over a valuable skill that was readily transferable from organization to organization—that provided leverage in the labor market.[63] 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 labor market fluctuations, and it was seen by many workers as a means of advancement into the middle class.[64] The 1960s were a period when many white-collar occupations were pursuing professional agendas, and the sociological literature of the period seemed to provide a clear road map to the benefits of professionalism. It appeared to many that these benefits were available to almost any occupation, assuming only that they followed the appropriate road map.[65]

The professionalization efforts of computer specialists were also encouraged, to a certain extent, by their corporate employers. Professionalism, or at least a specific form of corporate-friendly professionalism, provided a familiar solution to the increasingly complex problems of programmer management. Argued one personnel research journal from the early 1970s,

> The concept of professionalism 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.[66]

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."[67] At the very least, allowing programmers to think that they were professionals would go a long way toward reducing turnover and maintaining the stability of the data-processing staff.[68]

The widespread adoption of the rhetoric of professionalism conceals, however, deep intellectual and ideological schisms that existed within the programming community. Although many practitioners agreed on the need for a programming profession, they disagreed sharply about what such a profession should look like. What was the purpose of the profession? Who should be allowed to participate? Who would control entry into the profession, and how? What body of abstract knowledge would be used to support its claims to legitimacy? By the beginning of the 1960s, clearly discernible factions had emerged within the nascent programming discipline, each pursuing very different models of professional development.

Next, I focus on the professionalization efforts of two of the most prominent professional associations for computer personnel in this period, the Association for Computing Machinery (ACM) and the DPMA. The differences between the professional agendas advocated by each of these associations—the former based on the model of the research scientist, the latter on the certified public accountant—reveals the limits of professionalism as a solution to the "persistent personnel problem" of the late 1960s.

Computer science as the key to professionalism

By the late 1950s, numerous computer-related studies were ongoing in a variety of academic departments at various research universities, including departments of mathematics, business and economics, library science, physics, and electrical engineering. In a 1959 article, "The Role of the University in Computers, Data Processing, and Related Fields," Louis Fein argued that all these activities should be consolidated into a single organizational entity. He experimented with several names for this new entity, including information sciences, intellitronics, synnoetics, and computer science.[69] Other names for this new discipline (or its practitioners) had been suggested elsewhere in the contemporary literature: Comptology, Hypology (derived from the Greek root *hypologi*, meaning to compute), Applied Epistemologist, and Turingineer, among others. Computer science was the name that stuck.[70]

The academically oriented agenda outlined by Fein and other supporters of *computer science* provided a familiar model of professional advancement. The sociological literature of the era suggested that the key to professionalism was the control of abstract knowledge: The more theoretical the discipline, the greater its professional status and autonomy. "As a profession becomes mature it realizes that the science (not technology) needed by the profession must continually be extended to more basic content rather than restricted only to the obvious applied science," argued C.M. Sidlo in a 1961 essay, "The Making of a Profession."[71] The primary distinction between professionals and technicians, another observer suggested, "is based on whether one has undergone a 'prolonged course of specialized, intellectual instruction and study.'"[72] Without a "cohesive and consistent body of theory or theories" on which to base their discipline, computer science could "hardly be classified as a discipline demanding a separate curriculum and an isolated program."[73] By the end of the 1960s, some degree of formal education was seen by an increasing number of contemporary data-processing personnel as a necessary prerequisite to professional status. Exactly how much—and what kind—of formal education remained a point of considerable contention.

The move toward a more scientific approach to computing was greatly assisted by the support from the ACM's leadership. From its inception, the ACM styled itself as an academically oriented organization. Many of the original members either were or had been associated with a major university computation project, and most were university educated, a number at the graduate level. The association had been founded at an academic conference (the 1947 Symposium on

Large-Scale Digital Calculating Machinery at the Harvard Computation Laboratory), and the organization's early activities focused on a series of national conferences that retained a distinctly academic flavor. Many were low-budget affairs held at universities or research institutions and frequently made use of dormitory facilities (much to the dismay of industry participants on expense accounts). The papers presented were usually technical, and the proceedings were published. The ACM conferences never acquired the trade-show atmosphere that characterized other national meetings. In fact, deliberate efforts were made to distance the ACM from the influence of "all commercial considerations [including] the sale of publications and the solicitation of advertising."[74] Until 1953, when it began publishing the *Journal of the ACM*, the association's preferred forum for publications was the National Research Council's highly technical journal *Mathematical Tables and Other Aids to Computation*. Even then, the primary contents of the *Journal* were theoretical papers, and the emphasis was on disseminating "information about computing machinery in the best scientific tradition."[75] Articles were peer reviewed, and every attempt was made to maintain rigorous academic standards. As early as 1959, it had been proposed that the ACM impose stringent academic standards on its members, and in 1965 a four-year degree became a prerequisite for receiving full membership.

Throughout the 1950s and early 1960s, the ACM continued to cultivate its relationship with the academic community. It accepted an invitation in 1954 to apply for membership in the American Association for the Advancement of Science. Since 1958, the ACM has been represented in the Mathematical Sciences Division of the National Academy of Sciences' National Research Council. In 1962, it affiliated with the Conference Board of the Mathematical Sciences, which also consisted of the American Mathematical Society, the Mathematical Association of America, the Society for Industrial and Applied Mathematics, and the Institute of Mathematical Statistics. In 1966, the ACM established the prestigious Turing Award, the highest honor awarded in computer science. Almost half of the ACM's institutional members were educational organizations, and by 1962, a thriving student membership program had been developed.[76]

The close association that the ACM maintained with the academic community proved a mixed blessing, however. Although it provided the ACM a certain degree of authority in its claims for professional recognition, it also pulled the association in a direction that did not always appeal to more business-oriented computer personnel. A 1963 *Datamation* article, "The Cost of Professionalism," warned that

> They've [the members of the ACM] got to decide whether it's worth that much to belong to an organization which many feel has been dominated by—and catered pretty much to—Ph.D. mathematicians … the Association tends to look down its nose at business data processing types while claiming to represent the whole, wide wonderful world of computing.[77]

A 1966 Diebold Group publication characterized the ACM as a group "whose interests are primarily academic and which is helpful to those with scholastic backgrounds, theoreticians of methodology, scientific programmers and software people."[78] Although the ACM president immediately denied this characterization, calling it too narrow, the popular perception that the ACM catered solely to academics was difficult to counter.[79] Even among its supporters,

> the notion that ACM is a sort of holier than thou academic intellectual sort of enterprise—not inclined to be messing around with the garbage that comptrollers worry about—has been and still is, to a sufficiently large percentage of the members, an attitude that one can feel considerably paranoid about.[80]

The perception that academic computer scientists were too academic to be much use to professional programmers was in part an unfortunate consequence of contemporary academic politics. As William Aspray has suggested, computer science crossed virtually every academic boundary then established within the university, drawing content and people from mathematics, electrical engineering, psychology, and business.[81] As computer-related subfields began drawing resources and students from traditional disciplines, heated battles erupted over faculty slots, graduate admissions, and courses. Conflict between computer science and these older departments was almost inevitable. Its early success at attracting students and resources notwithstanding, computer science was repeatedly forced to defend its academic legitimacy.

Critics of computer science accused it of being little more than a grab bag of techniques, heuristics, and equipment. The discipline's close association with computer hardware was evoked to disparage its intellectual legitimacy:

> The creation of computer science departments is analogous to creating new departments for the railroad, automobile, radio, airplane or television technologies. These industrial developments were all tremendous innovations embodied in machinery, as is the development of computers, but this is not enough for a discipline or a major academic field.[82]

As Atsushi Akera has suggested, as personnel from university computing centers moved into newly founded computer science departments, they had difficulty shedding their image as service providers rather than legitimate researchers.[83] Computer science "is viewed by other disciplines as a rather easily mastered tool," computer theorist David Parnas warned an ACM curriculum committee in 1966.

> It is easy, in any field, to confuse the work of a technician with the work of a professional, but this is easier in computer science because a worker in another discipline will consider himself an 'expert' after learning to use a computer to process his data.[84]

The response of the academic computer science community to accusations of insufficient theoretical rigor was understandable: They focused increasingly on those aspects of their discipline that most resembled traditional science and mathematics. In his 1959 manifesto announcing the new discipline, for example, Louis Fein had been careful to distance the science of computing from its hardware-oriented origins:

> Too much emphasis has been placed on the computer equipment in university programs that include fields in the 'computer sciences' … Indeed an excellent integrated program in some selected fields of the computer sciences should be possible without any computing equipment at all, just as a first rate program in certain areas of physics can exist without a cyclotron.[85]

A 1964 report from the ACM Curriculum Committee on Computer Science echoed this notion that computer science involved more than just the design and operation of computing equipment: "Computer science is concerned with information in much the same sense that physics is concerned with energy."[86]

As Paul Ceruzzi has suggested, by the end of the 1960s most computing theorists had adopted the definition of their discipline that emphasized its theoretical aspects: computer science as the study of algorithms.[87] Implied in this definition is the notion that the algorithm is as fundamental to computing as Newton's laws of motion are to physics. By founding their discipline on the algorithm rather than on engineering practices, computer scientists could claim fellowship with the sciences: Computer science was science because it was concerned with discovering natural laws about algorithms.

This commitment to an abstract and theoretical approach to computing research and education was further reinforced the following year when the ACM Committee on Curriculum for Computer Science announced their Curriculum '68 guidelines for university computer science programs, which encouraged university computer science departments to drop electronics and hardware courses in favor of mathematics and algorithms offerings.[88] The Curriculum '68 report included little of interest to employers and business practitioners, particularly when compared to alternative programs advanced by the IEEE or the DPMA.[89] Even when the ACM did recognize the growing importance of business data processing to the future of its discipline, the emphasis was always placed on research and education:

> All of us, I am sure, have read non-ACM articles on business data processing and found them lacking. They suffer, I believe, from one basic fault: They fail to report fundamental research in the data processing field. The question of 'fundamentalness' is all-important … In summary, this letter is intended to urge new emphasis on FUNDAMENTALISM in business data processing. This objective seems not only feasible but essential to me. It provides not only a technique for getting ACM into the business data processing business, but a technique (the same one) for getting the field of business data processing on a firm theoretical footing.[90]

The ACM leadership was not entirely unaware of or unsympathetic to the needs of the business programmers, who often held different ideas about what constituted a computing professional. In his unsuccessful 1959 bid for the ACM presidency, Paul Armer urged the ACM membership to "THINK BIG," to "visualize ACM as the professional society unifying all computer users."[91] That same year, Herbert Grosch, an outspoken proponent of a strong, American Medical Association–style professional society, roundly criticized the ACM for its academic parochialism:

> Information processing is as broad as our culture and as deep as interplanetary space. To allow narrow interests, pioneering though they might

have been, to preempt the name, to relegate ninety percent of the field to 'an exercise left to the reader,' would be disastrous to the underlying unity of the new information sciences.[92]

Several attempts were made during the next decade to make the ACM more relevant to the business community. In response to widespread criticism of the *Journal of the ACM*'s theoretical orientation, a new publication—the *Communications of the ACM*—was introduced in 1958. The main contents of the *Communications* were short articles, mostly unrefereed, on technical subjects such as applications, techniques, and standards.[93] In 1966, the Executive Committee announced a $45,000 professional development program aimed at business data-processing personnel. The program includes short skill upgrade seminars offered at the national computer conferences, a traveling course series, and self-study materials.[94] There was even talk, in the mid-1960s, of a potential merger with the DPMA. In 1969, ACM president Bernard Galler announced a move toward "less formality, less science, and less academia."[95]

Although the majority of ACM members were from industry by the end of the 1960s, the conservative ACM leadership continued to pursue a largely academic agenda. Most of the ACM presidents from this period came from universities or other research institutions: Bernard Galler from the University of Michigan, Anthony Oettinger from Harvard, and Paul Armer from Rand. Many of the SIGs, which controlled a great deal of the money and power within the ACM, had an academic thrust. Given that leadership was so highly concentrated and centralized (particularly in contrast with competing associations like DPMA, which encouraged the development of strong local chapters), it was often difficult for the ACM to respond quickly to changes in the social and technical environment of computing. For example, frequent battles arose over repeated attempts to change the name of the association to something more broadly relevant. In 1965, a proposal to change it to the Association for Computing and Information Science was rejected; a decade later, the same issue was still being debated.[96] When Louis Fein suggested in a 1967 letter to the editors of the *Communications* that the ACM faced a "crisis of identity," ACM President Anthony Oettinger insisted vehemently that the "ACM has no crisis of identity." In doing so, he reaffirmed the association's commitment to a theoretical approach to computing: "Our science must, indeed, 'maintain its sole abstract purpose of advancing truth and knowledge.'"[97]

There is little question that throughout the 1960s the ACM pursued a professionalization strategy that was heavily dependent on the authority and legitimacy of its academic accomplishments. The model of professionalism advocated by academic computer scientists, however, did not always correspond with interests of their industry colleagues. The skills and abilities rewarded in the academy were not necessarily valued within the corporate environment:

> These four year computer science wonders are infinitely better equipped to design a new compiler than they are to manage a software development project. We don't need new compilers. We need on-time, on-budget, software development.[98]

Alternative models began to emerge for pursuing professional development in computing, models that focused more on the immediate needs of corporate employers and business data processing.

The certified public programmer

In 1962, the editors of *Datamation* issued a call for the creation of a new technical profession: the certified public programmer. The establishment of a rigorous certification program for computer personnel, they argued, would help resolve some of the "many problems" that were "embarrassingly prominent" in the contemporary software industry.[99] By defining clear standards of professional competency, an industry-wide certification program would serve several important purposes for the nascent programming profession. First, it would establish a shared body of abstract occupational knowledge—a "hard core of mutual understanding"—common across the entire professional community. Second, it would help raise the public's view of computing professionals "several impressive levels from its current stature of cautious bewilderment and misinterpretation to, at least, confused respect."[100] Finally, and perhaps most significantly, it would enable computer professionals to erect barriers to entry to their increasingly contested occupational territory:

> With a mounting tide of inexperienced programmers, new-born consultants, and the untutored outer circle of controllers and accountants all assuming greater technical responsibility, a need for qualification of competence is clearly apparent.[101]

The *Datamation* editorial on the need for professionalism within the computer industry coincided neatly with the announcement by the National Machine Accountants Association (NMAA) of their new Certificate in Data Processing (CDP) examination. The NMAA, which would later that year rename itself the DPMA, represented almost 16,000 data-processing workers in the US and Canada. The NMAA had been working since 1960 to develop the CDP exam, which represented the first attempt by a professional association to establish rigorous standards of professional accomplishment in the data-processing field. According to the association's 1962 press release, the exam was intended to "emphasize a broad educational background as well as knowledge of the field of data processing," and to represent "a standard of knowledge for organizing, analyzing and solving problems for which data processing equipment is especially suitable." It was open to anyone, NMAA member or not, who

- completed a prescribed course of academic study,
- had at least three years direct work experience in punched card and/or computer installations, and
- had "high character qualifications."[102]

The educational requirements were waived through the 1965 examinations, however. The first year it was offered, 1,048 applications took the CDP examination, and 687 passed.[103]

Although the CDP program was criticized by some as being overly broad and superficial, by the end of 1965 almost 7,000 programmers had sat for the exam, and the CDP appeared to be well on its way toward becoming a widely accepted industry certification standard. Large firms such as State Farm Insurance, the Prudential Insurance Company of America, and the US Army Corps of Engineers extended official recognition to the CDP program, and in 1964, the city of Milwaukee began using the CDP as a means to assign pay grades to data-processing personnel.[104] In 1966, the DPMA mounted a major promotional campaign for the CDP program that included press releases, radio advertisements, and television infomercials.[105] The striking early success of the program, which more than quintupled in size in its first three years, suggests that many programmers saw certification as an attractive professional strategy. This corresponds well with evidence from industry journals and other documentary sources. A survey of the 1963 candidates reveals a remarkable range of background, experience, and education.[106]

The DPMA's professionalization agenda was clearly derived from the model of the CPA. Deriving as it did from an association of accountants (the NMAA), many of its original members and leaders had backgrounds in accounting. The CDP program was only one of the DPMA's ambitious "Six Measures of Professionalism" program, which included additional provisions for programs in continuing education, public service, and the development of a professional code of ethics. Of the proposed measures, only the CDP program achieved even moderate industry acceptance; nevertheless, simply by articulating a clear professional agenda, the DPMA claimed for itself a leadership role in the computing community. From the beginning, the DPMA made efforts to reach a broad spectrum of computer specialists, including business programmers. The structure of the organization, which included strong regional chapters, allowed for diversity and local control (in marked contrast to the ACM). Its official publication, the *Data Management Journal*, encouraged submissions on a much wider range of subjects—including product reviews, practical tips and techniques, and even computer humor—than did the ACM's *Journal* or *Communications*. The DPMA also maintained a close association with the editors of *Datamation*, which focused on issues of timely concern and practical relevance.

The DPMA was not alone in its interest in establishing certification standards for data-processing personnel. Given the general lack of agreement on what skills and educational background were appropriate for data-processing personnel, certification programs promised the guarantee of at least a basic level of competence. Employers viewed certification as a means of screening potential employees, evaluating performance and awarding promotions, and assuring uniform product and quality.[107] Programmers saw it as an indication of professional status, as a means of assuring job security and achieving promotions, and as an aid to finding and obtaining a new position.[108] Furthermore, the certification of practitioners was considered one of the characteristic functions of any legitimate profession.[109] The establishment of a successful certification program was thought by many to be a precondition for professional recognition.

Although certification programs were appealing to many practitioners in theory, in actual practice they often proved unwieldy or even burdensome. Set the standards too low and the certification becomes meaningless; set them too high and risk excluding most of your primary constituency. For example, when the

DPMA first began to enforce the educational requirements included in the original 1962 CDP announcement, applications dropped by more than 85 percent, never to fully recover. The requirements were not terribly stringent—college-level courses in math, English, managerial accounting, statistics, and data-processing systems, as well as eight out of 17 possible electives—but many of the practicing EDP specialists who formed the core of the DPMA membership saw such requirements as being irrelevant, unattainable, or both.[110] A major controversy erupted within the pages of the industry journals, particularly the DPMA-oriented *Datamation* and *Computerworld*.

Advocates of the academic requirements argued that such requirements not only elevated the status and legitimacy of the CDP but were standard for most other professions, including law, medicine, and engineering. Opponents claimed that the specific course requirements were ambiguous, meaningless, and irrelevant. The DPMA Committee for Certification, which administered the CDP program, was flooded with letters from disgruntled applicants either complaining or requesting special dispensation. Faced with the imminent collapse of its membership support, the DPMA admitted that "the established eligibility requirements had unintentionally excluded some of the people for whom the CDP program was originally designed."[111] The Committee dropped the specific course requirements, providing a grandfather clause for those with three years' experience prior to 1965, and requiring others to have only two years of post-secondary education. Applications for the 1968 exam session jumped back to almost 3,000.

Over the next several years, the CDP program struggled to regain its initial momentum. Annual enrollments dropped again briefly in 1969, then leveled off for the next several years at about 2,700. In an industry characterized by rapid expansion, this noticeable lack of growth represented a clear failure of the CDP program. With each year, CDP holders came to represent a smaller and smaller percentage of the programming community.

In 1970, the program faced yet another crisis: the announcement that a baccalaureate degree would be required of all CDP candidates, beginning with the 1972 examination. Once again, a firestorm of debate broke out. The DPMA claimed that this new requirement merely reflected the changing reality of the labor market: Since a college degree had already become a de facto requirement within the industry, requiring anything less for the CDP would severely undermine its legitimacy. Nevertheless, the resulting controversy highlighted already existing tensions within the data-processing community, and it further divided the already fragmented DPMA Certification Council (many of whom could not themselves satisfy the new degree requirement). The head of the West Tennessee chapter of the DPMA wrote to complain that he, along with about one third of his chapter's membership, had suddenly become ineligible to receive the CDP. A 1970 *Computerworld* survey indicated that many practitioners felt the new requirement "unduly harsh" and "ludicrous," believing that it would decimate the data-processing staffs of many smaller departments. The always outspoken Herbert R. Grosch (himself a PhD astronomer and future ACM president) was quoted to the effect that "This policy is very ill-advised. What the hell is so hot about college—it turns out a bunch of knuckleheads—and a knucklehead PhD is no better than a knucklehead CDP."[112]

Despite the strong negative reaction generated by these educational requirements, the DPMA leadership continued to insist on their necessity. Such requirements had always been considered an essential component of the DPMA's professionalization program: Only by defining a "standard of knowledge for organizing, analyzing, and solving problems for which data processing equipment is especially suitable" could programmers ever hope to distinguish themselves from mere technicians or other "subprofessionals."[113] Like the academic computer scientists, business programmers recognized the need for a foundational body of abstract knowledge on which to construct their profession; they differed only on what that relevant foundation of knowledge should include. In insisting on strong educational standards, the DPMA was in complete accord with the conventional wisdom of the contemporary professionalization literature.[114]

The DPMA program also suffered greatly from a lack of support from other professional associations. A 1968 article on certification and accreditation in the *Communications of the ACM* entirely failed to mention the CDP. This conspicuous neglect of the most successful certification program then available reflected a larger pattern of hostility toward the DPMA on the part of the ACM leadership. In 1966, the Executive Council of the ACM considered a resolution, clearly aimed at the CDP, to "warn employers against relying on examinations designed for subprofessionals or professionals as providing an index of professional competence."[115] Later that year, they established the Committee to Investigate the Implications of the CDP, whose first order of business was drafting a strongly worded objection to the use of the word *professional* in association with the DPMA exam. The wording of subsequent exam and program literature eliminated all references to such language: CDP therefore came to stand for Certified Data Processor, rather than Certified Data Professional.[116] Even this more modest acronym was offensive to many critical observers.[117] Pressure from competing associations forced the DPMA to abandon many of its more ambitious claims for the CDP program.[118] A 1966 statement conceded that

> it would be presumptuous at this early stage in the program to suggest that CDP represents the assurance of competence, or that the Certificate should be considered as a requirement for employment or promotion in the field.[119]

It is no wonder that so many employers and practitioners lost confidence in the ability of the DPMA to successfully administer an industry-wide certification program.

It was this lack of confidence on the part of employers and practitioners that ultimately doomed the CDP program. The DPMA was unable to sustain widespread or lasting support from either group for their proposed educational standards or, for that matter, their certification exams. Neither was convinced that a CDP meant much as a measure of ability or future performance. The DPMA Certification Council was unable even to pass a resolution requiring its own officials to possess the CDP.[120] One of the major criticisms leveled against the CDP examination by employers and data-processing managers was that it tested "familiarity" rather than competence.[121] It was not clear to many observers what skills and abilities the CDP was actually intended to certify:

> The present DPMA examination measures breadth of data processing experience but does not measure depth … It certainly does not measure or qualify programming ability. It makes no pretense of being any measure of management skills.[122]

The problem was a familiar one for the industry: Although most employers in this period believed that only "competent" programmers could develop quality software, no one agreed on what knowledge and abilities constituted competence.[123] As Fred Gruenberger suggested at a 1975 Rand symposium on certification issues, "I have the fear that someone who has passed the certifying exams has either been certified in the wrong things (wrong to me, to be sure) or he has been tuned to pass the diagnostics, and in either case I distrust the whole affair."[124] His attitude reflected the ambivalence that many observers in this period felt about contemporary data-processing training and educational practices. If data processing was simply a "miscellaneous collection of techniques applied to business, technology and science," rather than a unique discipline requiring special knowledge and experience, then no certification exam could possibly test for the broad range of skills associated with "general business knowledge." "Given the choice between two people from the same school, one of whom has the CDP, but the other appears brighter," Gruenberger argued, "I'll take the brighter guy."[125]

In the absence of a strong commitment to the CDP on the part of employers, many programmers saw little benefit in participating in the program. Those who did were increasingly self-selected from the lowest ranks of the labor pool, individuals for whom the CDP was a perceived substitute for experience and education. By the mid-1970s, it became increasingly clear that the CDP program as it then existed faced imminent dissolution. In an attempt to restore momentum to their flagging certification initiative, the DPMA joined forces with seven other computing societies to form the Institute for Certification of Computer Professionals (ICCP). The ICCP never managed to revive the CDP or to institute a meaningful certification program of its own, however. Because it represented such a wide variety of constituents, the ICCP was hindered by the same internal divisions that plagued the larger programming community. Rivalries among the constituent member societies, many of whom were only superficially committed to the concept of certification, doomed the organization to internal conflict and inactivity.[126]

The limits of professionalism

The persistent conflict between the ACM and the DPMA reflected a much larger tension that existed within the computing community. As early as 1959, the outlines of a battle between business programmers and academically oriented computer scientists had taken shape around the issue of professionalism.[127] Although both groups agreed on the desirability of establishing institutional and occupational boundaries around the nascent computer-related professions, they disagreed sharply about what form these professional structures should take. Observers noted a deepening "programming schism" developing in the industry, a "growing breach between the scientific and engineering computation boys who talk Algol and Fortran … and the business data processing boys who talk English and write programs in Cobol."[128] Individuals who believed that the key to professional status was the development of formal theories of computer science resisted "subprofessional" certification programs and tended to join the ACM; business data processors who saw the ACM leadership "as a bunch of guys with their heads in the clouds worrying about Tchebysheff polynomials and things like that" either supported the DPMA or ignored the professional societies altogether.[129]

The inability of programmers and other data-processing personnel to successfully professionalize raises some perplexing questions for the historian: Given the apparent advantages of professionalization to both employers and practitioners, why were these efforts so ineffective? As was described earlier, industrial employers in the 1960s complained not as much about technical incompetence as a general lack of professionalism among programmers. The computer scientist Malcolm Gotterer, speaking of the "typical EDP specialists," suggested that

> It was his distressing lack of professional attributes that most often undermines his work and destroys his management's confidence. Too frequently these people, while exhibiting excellent technical skills, are nonprofessional in every other aspect of their work.[130]

Increased professionalism would presumably address the most frequent complaints leveled against data-processing personnel: an overreliance on idiosyncratic craft techniques, an arrogant disregard for proper lines of authority, shoddy workmanship, and a lack of commitment to the best interests of the organization. On the surface, the professionalization of programming appeared to be an ideal solution to many of the most deleterious symptoms of the burgeoning software crisis.

It is clear that the turf battles that raged between the ACM and the DPMA during the 1950s and 1960s helped undermine popular support for both organizations. In response to extensive *Datamation* coverage of a 1959 Rand symposium on "the perennial professional society question," one reader commented that he "hadn't laughed so hard in a decade. Are these guys kidding? You won't solve this problem by self-interested conversation about it, nor is it solved by founding another organization."[131] Many observers were dismayed by the pettiness of the ACM–DPMA debates, which they believed detracted from the overall goal of establishing a legitimate professional identity:

> I couldn't care less who publishes some abstract scientific paper! What I want to know is how do we pull together a hundred thousand warm bodies that are working on the outskirts of the computer business, give them a high priced executive director, lots of advertising, a whole series of technical journals; in other words, organize a real rip-snorting profession? Whenever somebody starts worrying about which journal what paper should be published in, we get bogged down in an academic cross-fire we've been in for ten years.[132]

As the programming community broke down into competing factions—theoretical versus practical, certified versus uncertified, ACM versus DPMA—its members lost the leverage necessary to push through any particular professionalization agenda.

In addition to interassociational rivalries, the aspiring computing professions also faced external opposition. For many corporate managers, professionalism was a potentially dangerous double-edged weapon. On the one hand, "Professionalism 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 determine 'who is qualified.'"[133] On the other hand, "professionalism might well increase staff mobility and hence turnover, and it probably would lead to higher salaries for the 'professionals.'"[134] Computer personnel were often seen as dangerously disruptive to the traditional corporate establishment. The last thing most traditional managers wanted was to provide data-processing personnel with additional occupational authority.[135] Professionalism was therefore encouraged only to the extent that it provided a standardized, tractable workforce; professionalization efforts that encouraged elitism, protectionism, or anything that smacked of unionism were seen as counterproductive.

Perhaps the most important reason that programmers and other data-processing personnel failed to professionalize, however, was that the professional institutions that were set up in the 1950s and 1960s failed to convince employers of their relevance to the needs of business. Employers looked to professional institutions as a means of supplying their demand for competent, trustworthy employees. As we have seen, although computer science programs in the 1960s thrived in the universities, in the business world they were often seen as overly theoretical and irrelevant. Likewise, the DPMA's CDP program failed to establish itself as a reliable mechanism for predicting programmer performance or ability. Neither the ACM nor the DPMA offered much to employers in terms of improving the supply or quality of the programming workforce.

Given this lack of active support from employers, the professional associations had little to offer most data-processing practitioners. Neither a computer science education nor professional certification could ensure employment or advancement. As a result, many computer personnel saw little value in belonging to either the ACM or the DPMA, and support for both organizations, as well as for professional institutions in general, languished during the late 1960s and early 1970s. A 1967 *Datamation* article indicated that "Less than 40% [of programmers] belong to any professional association. Probably less than 1% do anything in connection with an association that requires an extra effort on the individual's part."[136] Even these low figures were probably inflated: A *Wall Street Journal* report from the next year revealed only that 13 percent of the data-processing personnel surveyed belonged to any professional society.[137]

Conclusions

In the decades following the 1960s, the "question of professionalism" has remained one of the dominant themes in the computer industry literature. The May 2000 issue of the IEEE magazine *Computer* was almost entirely devoted to the subject, for example.[138] The ongoing debate about the nature and causes of the so-called software crisis, which is now in its fourth decade, has always been intimately tied up with the discussion about professionalism in the computing disciplines.[139] A recent book on software professionals noted a list of barriers to professionalism in the computer fields surprisingly similar to those identified in the early 1960s: an influx of untrained personnel, the lack of established educational and certification standards, and the absence of abstract "foundational disciplinary principles."[140]

The purpose of this article has not been to address the question of whether computing is a real profession. Indeed, the current literature on the sociology of the professions suggests that this question is largely irrelevant: Even the paradigmatic professions of law and medicine are difficult to categorize using the simplistic structural criteria developed in the 1960s.[141] My intention has simply been to demonstrate the rich complexities hidden behind seemingly straightforward debates about professionalism. Like the "worldwide shortage of information technology workers" of the current era, the "acute shortage of programmers" of the 1960s was about more than a mere disparity between supply and demand.[142] The problem was not so much a lack of computer specialists per se but rather the lack of a certain kind of computer specialist. Teasing apart just what that certain kind of specialist was supposed to be goes a long way toward understanding the larger social and political context of these debates.

The approach to the institutional and organizational history of the computing professions outlined here suggests new interpretations of a number of important episodes in the history of computing. For example, one of the most intriguing and influential developments in the history of software has been the widespread adoption of the rhetoric and ideology of software engineering. When Presper Eckert first introduced the concept in 1965, it received little attention from industry pundits.[143] Just several years later, however, the 1968 NATO Conference on Software Engineering firmly entrenched the language of software engineering in the vernacular of the computing community, thereby setting an agenda that influenced many of the technological, managerial, and professional developments in commercial computing for the next several decades. Why did software engineering suddenly emerge as such a compelling model for professional development? What did being a software engineer offer aspiring computer professionals that being a computer scientist or certified public programmer did not? Why did the software engineering model appeal to employers? How did other corporate employees—and traditional engineers—respond to this new professional self-definition? The full history of software engineering as a profession is obviously beyond the scope of this article. The issues and approaches identified here should, however, provide a context for understanding this and other crucial developments in the history of the computing professions. The debate that took shape in the 1950s and 1960s about who qualified to be a computer professional was about more than just the structures of professionalism: It represented a series of highly contested social negotiations about the role of electronic computing—and of computing professionals—in modern corporate and academic organizations.

References and notes

1. H.A. Rhee, *Office Automation in Social Perspective: The Progress and Social Implications of Electronic Data Processing*, Basil Blackwell, Oxford, 1968, p. 118.

2. E. Dijkstra, "The Humble Programmer," in *ACM Turing Award Lectures: The First Twenty Years, 1966–1985* ACM Press, New York, 1987. When Dijkstra applied for a marriage license in his native Holland, some years earlier, he had been rejected on the grounds that he had listed his occupation as programmer, which was not a recognized, legitimate profession. Dijkstra swallowed his pride, as he tells the story, and resubmitted his application with his "second choice"—theoretical physicist.
3. W. Aspray, "The History of Computer Professionalization in America," (unpublished manuscript).
4. C.J.A., "In Defense of Programmers," *Datamation*, vol. 13, no. 9, 1967; "Editor's Readout: The Certified Public Programmer," *Datamation*, vol. 8, no. 3, Mar. 1962, p. 15.
5. R. Canning, "Professionalism: Coming or Not?," *EDP Analyzer*, vol. 14, no. 3, Mar. 1976, p. 8. Canning wrote a series of articles on professionalism in the late 1960s and early 1970s. For examples, see also "The Question of Professionalism," *EDP Analyzer*, vol. 6, no. 12, Dec. 1968, pp. 1-13; "The Persistent Personnel Problem," *EDP Analyzer* vol. 5, no. 5, May 1967, pp. 1-14; "Career Programs in Data Processing," *EDP Analyzer*, vol. 9, no. 8, Aug. 1971.
6. J. Golda, "The Effects of Computer Technology on the Traditional Role of Management," master's thesis, Wharton School, Univ. of Pennsylvania, 1965, p. 34.
7. For example, see T. Alexander, "Computers Can't Solve Everything," *Fortune*, Oct. 1969, p. 169, and T. Whisler, "The Impact of Information Technology on Organizational Control," *The Impact of Computers on Management*, Charles Myers, ed., MIT Press, Cambridge, Mass., 1967, p. 44.
8. W.R. Walker, "MIS Mysticism," (letter to editor), *Business Automation*, vol. 16, no. 7, July 1969, p. 8.
9. R. Rosin, "Relative to the President's December Remarks," *Comm. ACM*, vol. 10, no. 6, June 1967.
10. A.W. Jacobson, ed., *Proc. First Conf. Training Personnel for the Computing Machine Field* held at Wayne Univ., Detroit, Mich., 22-23 June 1954, Wayne Univ. Press, Detroit, 1955. Quotation is from p. 79.
11. W.H. Wilson of the General Motors Corporation, quoted in Jacobson, p. 21.
12. T.C. Rowan, "The Recruiting and Training of Programmers," *Datamation*, vol. 4, no. 3, Mar. 1958. SDC was the Rand Corporation spin-off company responsible for developing the software for the SAGE air-defense system. In the late 1950s, the personnel management department at SDC trained more than 2,000 programmers, effectively doubling the number of trained programmers in the US.
13. R. Patrick, "The Gap in Programming Support," *Datamation*, vol. 7, no. 5, May 1961.
14. "Editor's Readout: A Long View of a Myopic Problem," *Datamation*, vol. 8, no. 5, May 1962, p. 21.
15. R. Tanaka, "Fee or Free Software," *Datamation*, vol. 13, no. 10, Oct. 1967, pp. 205-206.
16. American Federation of Information Processing Societies, "The State of the Information Processing Industry," a report commissioned for the Council for Economic and Industrial Research and presented at the 1966 Spring Joint Computer Conf.
17. G. Bylinsky, "Help Wanted: 50,000 Programmers," *Fortune*, Mar. 1967, p. 141.
18. J. Saxon, "Programming Training: A Workable Approach," *Datamation*, vol. 9, no. 12, Dec. 1963, p. 48.
19. R. Canning, "The Persistent Personnel Problem."
20. E. Markham, "Selecting a Private EDP School," *Datamation*, vol. 14, no. 5, May 1968.
21. L. Mandel, "The Computer Girls," *Cosmopolitan*, Apr. 1967.
22. News Brief, "First Programmer Class at Sing Sing Graduates," *Datamation*, vol. 14, no. 6, June 1968.
23. H. Sackman, "Conference on Personnel Research," *Datamation*, vol. 14, no. 7, July 1968.
24. A. Orden, "The Emergence of a Profession," *Comm. ACM*, vol. 10, no. 3, Mar. 1967, p. 146.
25. "Professionalism Termed Key to Computer Personnel Situation," p. 156.
26. P. Randall, "Need for Warm Bodies," *Datamation*, vol. 9, no. 10, Oct. 1963, p. 14.
27. J. Callahan, "To the editor ... ," *Datamation*, vol. 7, no. 3, Mar. 1961, p. 7.
28. E. Menkhaus, "EDP: Nice Work If You Can Get It," *Business Automation*, Mar. 1969, pp. 41-45, 74.
29. R. Hamming, "One Man's View of Computer Science," *ACM Turing Award Lectures: The First Twenty Years, 1966–1985*, ACM Press, New York, 1987.
30. W. Paschell, *Automation and Employment Opportunities for Office Workers; A Report on the Effect of Electronic Computers on Employment of Clerical Workers*, Bureau of Labor Statistics, Washington, D.C., 1958, p. 11.
31. *Proc. Rand Symp.*, 1969, Charles Babbage Inst. Archives, CBI 78, Box 3, Fld. 4, Univ. of Minnesota, Minneapolis.
32. This seems to be as true in the 1990s as it was in the 1960s. See, for example, W. Gibbs, "Software's Chronic Crisis," *Scientific American*, Sept. 1994, p. 86.
33. E. Markham, "EDP Schools—An Inside View," *Datamation*, vol. 14, no. 4, Apr. 1968, p. 22.
34. Editorial, "Editor's Readout: The Certified Public Programmer," *Datamation*, vol. 8, no. 3, Mar. 1962.
35. D. Ross, "Certification and Accreditation," *Datamation*, vol. 14, no. 9, Sept. 1968.
36. L. Fulkerson, "Should There Be a CS Undergraduate Program?," (letter to editor), *Comm. ACM*,

vol. 10, no. 3, Mar. 1967.
37. D. McCracken, "The Human Side of Computing," *Datamation*, vol. 7, no. 1, Jan. 1961, p. 10.
38. C.J.A., "In Defense of Programmers," p. 15.
39. H. Grosch, "Computer People and their Culture," *Datamation*, vol. 7, no. 10 Oct. 1961, p. 51.
40. ———, "The Computer Personnel Research Group," *Datamation*, vol. 9, no. 1 Jan. 1963, p. 38.
41. L. Kaufman and R. Smith, "Let's Get Computer Personnel on the Management Team," *Training and Development J.*, Dec. 1966, pp. 25-29.
42. Canning, "Career Programs in Data Processing."
43. System Development Corp. (advertisement), *Comm. ACM*, vol. 3, no. 5, May 1960, p. A10.
44. Xerox Corp. (advertisement), *Datamation*, vol. 14, no. 4, Apr. 1968.
45. Mitre Corp. (advertisement), *Datamation*, vol. 12, no. 6, June 1966.
46. Bendix Computer Corp. (advertisement), *Datamation*, vol. 8, no. 9, Sept. 1962.
47. J. Jenks, "Starting Salaries of Engineers are Deceptively High," *Datamation*, vol. 13, no. 1, Jan. 1967.
48. Editorial, "Learning a Trade," *Datamation*, vol. 12, no. 10, Oct. 1966, p. 21.
49. For example, see C.I. Keelan, "Controlling Computer Programming," *J. Systems Management*, Jan. 1969; D. Herz, *New Power for Management*, McGraw-Hill, New York, 1969; R. Canning, "Managing the Programming Effort," *EDP Analyzer*, vol. 6, no. 6, June 1968, pp. 1-15; C. Lecht, *The Management of Computer Programming Projects*, American Management Assoc., New York, 1967.
50. M. Gotterer, "The Impact of Professionalization Efforts on the Computer Manager," *Proc. 1971 ACM Ann. Conf.*, ACM Press, New York, 1971, p. 368.
51. Fred Gruenberger noted this tendency as early as 1962: "You know, I've never seen a hot dog language come out yet in the last 14 years … that didn't have tied to it the claim in its brochure that this one will eliminate all programmers." His quote appeared in the Rand Symp., "On Programming Languages, Part II," *Datamation*, vol. 8, no. 11, Nov. 1962.
52. P. Naur, B. Randall, and J.N. Buxton, eds., *Software Engineering: Proc. NATO Conf.*, Petrocelli/Carter, New York, 1976, p. 89.
53. R. Canning, "Issues in Programming Management," *EDP Analyzer*, vol. 12, no. 4, Apr. 1974.
54. B. Rothery, *Installing and Managing a Computer*, Business Books, London, 1968, p. 80.
55. A.P. Ershov, "Aesthetics and the Human Factor in Programming," *Comm. ACM,* vol. 15, no. 7, July 1972, p. 502.
56. Ibid., p. 502.
57. In the early 1970s, several books expressing concerns about the "industrialization" of programming appeared, the most notable of which is P. Kraft, *Programmers and Managers: The Routinization of Computer Programming in the United States*, Springer-Verlag, New York, 1977.
58. Although the specific composition of the group changed from year to year, the attendees always represented the highest levels of leadership in the discipline: award-winning computer scientists, successful business entrepreneurs, association presidents, and prolific authors.
59. *Proc. Rand Symp.*, "Defining the Problem, Part II," *Datamation*, vol. 11, no. 9, Sept. 1965, pp. 23-35.
60. DPMA report, *Six Measures of Professionalism*, Charles Babbage Inst. Archives, CBI 88, Box 21, Fld. 40., Univ. of Minnesota, Minneapolis.
61. A. Oettinger, "ACM Sponsors Professional Development Program (President's Letter to ACM Membership)," *Comm. ACM*, vol. 9, no. 10, Oct. 1966, pp. 712-713.
62. H. Wilensky, "The Professionalization of Everyone?" *American J. Sociology*, vol. 70, no. 2, Feb. 1964, pp. 137-158.
63. M.S. Larson, *The Rise of Professionalism: A Sociological Analysis*, Univ. of California Press, Berkeley, 1977.
64. R. Zussman, *Mechanics of the Middle Class: Work and Politics Among American Engineers*, Univ. of California Press, Berkeley, 1985.
65. 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 Wilensky, "The Professionalization of Everyone?" (1964).
66. "Professionalism Termed Key to Computer Personnel Situation," pp. 156-157.
67. R. Canning, "Professionalism: Coming or Not?," p. 2.
68. R. Gordon, "Personnel Selection," *Data Processing—Practically Speaking*, F. Gruenberger and S. Naftaly, eds., Data Processing Digest, Los Angeles, 1967, pp. 85, 87.
69. L. Fein, "The Role of the University in Computers, Data Processing, and Related Fields," *Comm. ACM*, vol. 2, no. 10, Oct. 1959.
70. Q. Correll, "Letters to the Editor," *Comm. ACM*, vol. 1, no. 7, July 1958, p. 2; P.A. Zaphyr, "The Science of Hypology," (letter to editor), *Comm. ACM*, vol. 2, no. 1, Jan. 1959, p. 4; Editors of DATA-LINK, "What's in a Name?," (letter to editor), *Comm. ACM*, vol. 1, no. 4, Apr. 1958, p. 6. A more complete treatment of this history can be found in Paul Ceruzzi, "Electronics Technology and Computer Science, 1940–1975: A Coevolution," *Annals of the History of Computing*, vol. 10, no. 4, Apr. 1989, pp. 257-275.
71. C.M. Sidlo, "The Making of a Profession," (letter to editor), *Comm. ACM*, vol. 4, no. 8, Aug. 1961.
72. M. Gotterer, "The Impact of Professionalization Efforts on the Computer Manager," *Proc. 1971*

ACM Ann. Conf., ACM Press, New York, 1971, pp. 371, 372.

73. J. Carlson, "On Determining CS Education Programs," (letter to editor), *Comm. ACM*, vol. 9, no. 3, Mar. 1966.
74. E. Weiss, "Publications in Computing: An Informal Review," *Comm. ACM*, vol. 15, no. 7, July 1972.
75. S. Gass, "ACM Class Structure," (letter to editor), *Comm. ACM*, vol. 2, no. 5, May 1959, p. 4.
76. Charles Babbage Inst. Archives, CBI 88, Box 22, Fld. 1; Charles Babbage Inst. Archives, CBI 23, Box 1, Fld. 15; Univ. of Minnesota, Minneapolis.
77. Editorial, "The Cost of Professionalism," *Datamation*, vol. 9, no. 10, Oct. 1963.
78. A. Oettinger, "On ACM's Responsibility (President's Letter to ACM Membership)," *Comm. ACM*, vol. 9, no. 8, Aug. 1966.
79. Ibid., p. 546.
80. *Proc. Rand Symp*., 1969. Charles Babbage Inst. Archives, CBI 78, Box 3, Fld. 4; Univ. of Minnesota, Minneapolis.
81. W. Aspray, "Was Early Entry a Competitive Advantage?," *Annals of the History of Computing*, vol. 22, no. 3, July-Sept. 2000, p. 65.
82. J. Carlson, "On Determining CS Education Programs," (letter to editor), *Comm. ACM*, vol. 9, no. 3, Mar. 1966, p. 135.
83. The best available source on this material is A. Akera, *Calculating a Natural World: Scientists, Engineers and Computers in the United States, 1937–1968*, doctoral dissertation, History & Sociology of Science Dept., Univ. of Pennsylvania, 1998.
84. D. Parnas, "On the Preliminary Report of C3S," (letter to editor), *Comm. ACM*, vol. 9, no. 4, Apr. 1966, pp. 242-243.
85. T. White, "The 70's: People," *Datamation*, vol. 16, no. 7, July 1973, p. 11.
86. ACM Curriculum Committee, "An Undergraduate Program in Computer Science—Preliminary Recommendations," *Comm. ACM*, vol. 8, no. 9, Sept. 1965, p. 544.
87. For example, in his 1968 classic, *Fundamental Algorithms*, computer scientist Donald Knuth attempted to situate "the art of programming" on a firm foundation of mathematical principles and theorems. See also Paul Ceruzzi, "Electronics Technology and Computer Science."
88. ACM Curriculum Committee, "Curriculum 68: Recommendations for Academic Programs in Computer Science," *Comm. ACM*, vol. 11, no. 3, Mar. 1968, pp. 151-157.
89. R. Wishner, "Comment on Curriculum 68," *Comm. ACM*, vol. 11, no. 10, Oct. 1968; Datamation Report, "Curriculum 68," *Datamation*, vol. 14, no. 5, May 1968; Hamming, "One Man's view of Computer Science."
90. J. Postley, "Letter to Editor," *Comm. ACM*, vol. 3, no. 1, Jan. 1960.
91. P. Armer, "Thinking Big," (letter to editor), *Comm. ACM*, vol. 2, no. 1, Jan. 1959. Emphasis mine.
92. H. Grosch, "Plus and Minus," *Datamation*, vol. 5, no. 6, June 1959.
93. R. Payne, "Reaction to Publication Proposal," (letter to editor), *Comm. ACM*, vol. 8, no. 1, Jan. 1965.
94. A. Oettinger, "ACM Sponsors Professional Development Program (President's Letter to ACM Membership)," *Comm. ACM*, vol. 9, no. 10, Oct. 1966.
95. B. Galler, "The Journal (President's Letter to ACM Membership)," *Comm. ACM*, vol. 12, no. 2, Feb. 1969.
96. ——-, "Will You Vote for an Association Name Change to ACIS?," *Comm. ACM*, vol. 8, no. 7, July 1965; "Vote on ACM Name Change," (1978), Charles Babbage Inst. Archives, CBI 43, Box 3, Fld. 10, Univ. of Minnesota, Minneapolis.
97. A. Oettinger, "President's Reply to Louis Fein," *Comm. ACM*, vol. 10, no. 1, Jan. 1967.
98. G. DiNardo, "Software Management and the Impact of Improved Programming Technology," *Proc. 1975 ACM Ann. Conf.*, ACM Press, New York, 1975, pp. 288-289.
99. "Editor's Readout: The Certified Public Programmer," p. 23.
100. Ibid., p. 23.
101. Ibid., p. 23.
102. In response to criticism from the many otherwise qualified programmers who did not have formal mathematical training or college-level degrees, the educational requirements were suspended until 1965. The other prerequisites—three years' experience and "high character qualifications"—were so vague as to be almost meaningless and seem to have been only selectively enforced.
103. "Certificate in Data Processing," *Datamation*, vol. 9, no. 8, Aug. 1963.
104. DPMA Certificate Panel (1964), Charles Babbage Inst. Archives, CBI 46, Box 1, Fld. 17, Univ. of Minnesota, Minneapolis.
105. Charles Babbage Inst. Archives, CBI 116, Box 1, Fld. 10, Univ. of Minnesota, Minneapolis.
106. CDP Advisory Council, "Minutes of the Third Annual Meeting," 17–18 Jan. 1964, Charles Babbage Inst. Archives, CBI 88, Box 2, Fld. 3, Univ. of Minnesota, Minneapolis.
107. R, Canning, "The Question of Professionalism," p. 1.
108. R. Canning, "The DPMA Certificate in Data Processing," *EDP Analyzer*, vol. 3, no. 7, July 1965.
109. Sidlo, "The Making of a Profession," p. 366.
110. Datamation Report, "Certificate in Data Processing," *Datamation*, vol. 9, no. 8, Aug. 1963.
111. "DPMA Revises CDP Test Requirements," *Data Management*, Aug. 1967, pp. 34-35.
112. *Computerworld*, 19 Aug. 1970, Charles Babbage Inst. Archives, CBI 116, Box 1, Fld. 27, Univ. of

Minnesota, Minneapolis.
113. A. Orden, "The Emergence of a Profession."
114. C.M. Sidlo, "The Making of a Profession," p. 366.
115. From the DPMA file, "Notes on ACM (1966)." Charles Babbage Inst. Archives, CBI 46, Box 1, Fld. 3, Univ. of Minnesota, Minneapolis. An early draft of this document referred specifically throughout to the "DPMA certification program." Although the final version referred only to certification programs in the abstract, the target of its attacks was obviously the CDP.
116. DPMA Board of Directors, "Minutes of 10th meeting," 1966; Charles Babbage Inst. Archives, CBI 88, Box 2, Fld. 7, Univ. of Minnesota, Minneapolis.
117. Letter from Jack Yarbrough, Charles Babbage Inst. Archives, CBI 46, Box 1, Fld. 17, Univ. of Minnesota, Minneapolis.
118. L. Johnson, "Letter to Richard Kornblum," Charles Babbage Inst. Archives, CBI 46, Box 1, Fld. 16, Univ. of Minnesota, Minneapolis.
119. R. Calvin Elliot, "Editor's Page," *Data Management,* Feb. 1966, Charles Babbage Institute Archives, CBI 46, Box 1, Fld. 3, Univ. of Minnesota, Minneapolis.
120. "Executive Meeting Summary," 1966; Charles Babbage Inst. Archives, CBI 46, Box 1, Fld. 3 , Univ. of Minnesota, Minneapolis.
121. R. Canning, "The DPMA Certificate in Data Processing."
122. R. Canning, "The DPMA Certificate in Data Processing." Similar comments were made by Jack Yarbrough, Charles Babbage Inst. Archives, CBI 46, Box 1, Fld. 17, Univ. of Minnesota, Minneapolis.
123. M. Stone, "In Search of an Identity," *Datamation*, vol. 18, no. 3, Mar. 1972, p. 53.
124. *Proc. Rand Symp.*, "Problems of the AFIPS Societies Revisited," 1975, Charles Babbage Inst. Archives, CBI 78, Box 3, Fld. 7. , Univ. of Minnesota, Minneapolis.
125. Ibid., pp. 22-23.
126. P. Armer, "Editor's Readout: Suspense Won't Kill Us," *Datamation*, vol. 19, no. 6, June 1973.
127. *Proc. Rand Symp.*, "Is It Overhaul or Trade-in Time? Part II," *Datamation*, vol. 5, no. 5, May 1959.
128. C. Shaw, "Programming Schisms," *Datamation*, vol. 8, no. 9, Sept. 1962, p. 32.
129. *Proc. Rand Symp.*, 1969. Charles Babbage Inst. Archives, CBI 78, Box 3, Fld. 4, Univ. of Minnesota, Minneapolis.
130. M. Gotterer, "The Impact of Professionalization Efforts on the Computer Manager," p. 368.
131. W. Flywheel, "Letter to the Editor (On Professionalism)," *Datamation*, vol. 5, no. 5, May 1959, p. 2. The "other organization" that he was referring to was the American Federation of Data Processing Societies (AFIPS).
132. *Proc. Rand Symp.*, "Is It Overhaul or Trade-in Time? Part II," *Datamation*, vol. 5, no. 4, Apr. 1959, p. 27.
133. R. Canning, "Professionalism: Coming or Not?," p. 2.
134. Ibid., p. 2.
135. For a fuller discussion of this, see N. Ensmenger, *From 'Black Art' to Industrial Discipline: The Software Crisis and the Management of Programmers*, doctoral dissertation, History & Sociology of Science Dept., Univ. of Pennsylvania, 2001.
136. R. Jones, "A Time to Assume Responsibility," *Datamation*, vol. 13, no. 9, Sept. 1967, p. 160.
137. "Survey on Use of Service Bureaus," *Wall Street J.*, special report, 1969, Charles Babbage Inst. Archives, CBI 88, Box 30, Fld. 29, Univ. of Minnesota, Minneapolis.
138. T. Lethbridge, "What Knowledge is Important to a Software Professional?" and G. Pour, M. Griss, and M. Lutz, "The Push to Make Software Engineering Respectable," *Computer* vol. 33, no. 5, May 2000, pp. 35-51.
139. N. Ensmenger, "Software as Labor Process," in *Mapping the History of Computing: Software Issues*, U. Hashagen, R. Keil-Slawik, and A. Norberg, eds., Springer-Verlag, New York, to be published in 2002.
140. S. McConnell, *After the Gold Rush: Creating a True Profession of Software Engineering*, Microsoft Press, Redmond, Wash., 1999.
141. See, for example, A. Abbott, *The Systems of Professions: An Essay on the Division of Expert Labor*, Univ. of Chicago Press, Chicago, 1988.
142. The heated recent debate about a potential IT worker shortage reveals the highly contested nature of questions of skill, education, and certification. For a good introduction to this debate, see P. Freeman and W. Aspray, *The Supply of Information Technology Workers in the United States*, Computing Research Assoc., Washington, D.C., 1999.
143. In fact, the only published reference I could find to Eckert's speech was an offhand comment made by Robert Gordon in his "Review of Charles Lecht, The Management of Computer Programmers," *Datamation*, vol. 14, no. 4, Apr. 1968.

Nathan Ensmenger is a lecturer in the History and Sociology of Science Dept. at the University of Pennsylvania. He is currently working on a history of the development of the software engineering disciplines. Contact him at History and Sociology of Science Dept., University of Pennsylvania, Logan Hall, 249 S. 36th Street, Philadelphia, PA 19104-6304; (215)898-8697; nathanen@sas.upenn.edu.