

Computing the Profession

Peter J. Denning
Computer Science Department, 4A5
George Mason University
Fairfax, VA 22030
703-993-1525
pjd@gmu.edu

DRAFT 8/12/98

Revised 9/20/98

© Copyright 1998 by Peter J. Denning. You may make one copy for noncommercial personal use. Further copies or distributions require advance permission.

This essay has been prepared for the book Computer Science and Engineering Education, Tony Greening, Editor, and for Educom Review, John Gehl, Editor.

To most of the hundred millions of computer-users around the world, the inner workings of a computer are an utter mystery. Opening the box holds as much attraction as lifting the hood of a modern car. Users expect computing professionals to help them with their needs for designing, locating, retrieving, using, configuring, programming, maintaining, and understanding computers, networks, applications, and digital objects. They expect academic computer science to educate and train computing professionals, to be familiar with the changing technologies, and to maintain research programs that contribute to these ends. Students of computing look to faculty for a comprehensive, up-to-date view of a world with many fragments, for making sense of rapidly changing technologies, for assistance in framing and answering important questions, and for training in effective professional practices.

There are today are many branches of computing and information technology. These include traditional computer science, information systems, information science, software engineering, computer engineering, database engineering, network engineering, systems engineering, software architecture, human-computer interface design, computational science, computational statistics, numerical modeling, library

sciences, and several more. They share a common intellectual core but have different professional practices and concerns. Some of the direct offspring of computer science propose to split off into their own disciplines, while some of the newcomers propose to merge with computer science. Taken together, these groups constitute the emerging Profession of Computing.

Traditional computer scientists face a dilemma. Should they hold a conservative view, insisting that their offspring not separate and the newcomers not merge? If so, they run the risk of being sidelined in the new profession. Should they seek a leadership position in the new profession? If so, they must cross a chasm separating their current concerns from those of the multitude of clients who seek their expertise. To cross the chasm, they must embrace the birth of a new profession.

Crossing the Chasm

Computer scientists are known as independent, inventive, visionary, and proud. They have been criticized for being insular and disdainful of applications. They are no longer the primary inventors of hardware and software. They find themselves challenged by a multitude of users with mundane, practical concerns about using and relying on computers. Computer scientists, it seems, hardly have any influence over the direction of the technology any more. What role will they play in the new Profession of Computing?

I believe that computer scientists are experiencing a phenomenon described eloquently by Geoffrey Moore earlier in the decade [moor91]. No relation to Gordon Moore (the Intel founder famous for the 18-month doubling law of processor power), Geoffrey Moore is a principal of the Regis McKenna advertising agency headquartered in Silicon Valley. Moore had witnessed hundreds of new companies start life with marvelous inventions and rapid early market growth -- only to collapse suddenly within three years or their first \$20 million of expenditures. Their sales leveled or plummeted and they went out of business. They did not know what happened to them.

But Moore did. He explained the phenomenon and offered advice for those planning new companies. He recalled an earlier model of mindsets toward technologies, which divided people into five groups: the inventors, the visionaries, the pragmatists, the conservatives, and the luddites. Each successive group takes longer to grasp the implications of the new technology and to be sold on its use. Moore suggested that the distribution of people among categories follows a bell curve, meaning that the pragmatists are by far the largest group. The founders of companies are often inventors working in concert with visionaries. The founders meet initial success by selling their technology to other inventors and visionaries, who are quick to grasp the implications of the technology. But their downfall comes when they fail to persuade pragmatists to purchase their technology. The pragmatists worry about stability, dependability, and reliability; they want to use the technology but don't want to be victimized by breakdowns or held hostage by single suppliers. Moore invokes the metaphor of a chasm: the company leadership

discover too late that their marketing story and approach communicates with other early-adopters like themselves, but not with pragmatists. They do not have the resources or expertise to build the bridge. And so they go out of business.

Computing scientists (and other information technologists) are the inventors and visionaries in Moore's model. The multitudes of new users are pragmatists, whose concerns and demands differ sharply from those of early-adopters. Computing scientists thus face a chasm separating the world they know from the world in which computers are going to thrive in the future. To cross the chasm, they must embrace the emerging Profession of Computing.

The chasm between scientists and citizens who live and work with technology extends much further than computing. Science journalist Takashi Tachibana says that the chasm between scientists and non-scientists has widened during the 20th Century into a gulf. Unless scientists can find ways to communicate effectively with the multitudes, the basic research enterprise feeding technological development will dry up [tach98].

Struggles in the Growth of Computing

Moore's model suggests a growth process in which an organization gradually expands to larger markets. In reality, the stages of growth are not so well defined and have no sharp transition points. The discipline of computing illustrates this well. Computer science has been subject to demands from pragmatists for a long time and has struggled across several small chasms along the way. Those struggles have broadened the discipline and have helped prepare it for the new profession. The outcomes of earlier struggles have shaped how computer scientists approach the large chasm they face today. Who said crossing a chasm is easy?

Computer science boasts strong historical roots in engineering, mathematics, and science. The science roots, dating back to Galileo, reflect ancient interests in discovering the laws of nature and verifying them through calculation in many fields including astronomy, physics, and chemistry. The engineering roots, dating back to Michelangelo, reflect interests to harness the laws of nature through construction of artifacts and systems; in this century, electrical and electronic systems have been especially influential. The mathematics roots reflect interests in general methods (algorithms) for mechanically solving classes of problems and for characterizing rules of deduction -- e.g., Pascal in the 17th century, Gauss in the 18th, Hilbert in the 19th, Gödel and Turing in the 20th.

People from these three backgrounds came together in the 1940s to build the first electronic computers. While they cooperated freely, they also retained their identities in their fields of origin. There was much talk in the early days that the fledgling discipline of computer science might be a fad that would be reabsorbed into mathematics, electrical engineering, or physics. During its formative years, the discipline of computing had to contend with these built-in tensions.

At three times the interests of pragmatists intruded on the world created by the academic inventors and visionaries of the discipline. In the late 1970s, the field experienced a “brain drain” to industry of systems-oriented faculty, from which it never fully recovered. (A new brain drain appeared in the late 1990s with the rapid expansion of public interest in computing.) In the late 1980s, apathy toward computational science nearly led to the split-off of an important segment of the discipline. In the mid 1990s, ambivalence toward applications and engineering induced some software engineers to propose a separate discipline.

Experimental Computer Science

Experimental methods are dear to the heart of several core areas of computing, most notably the systems areas (e.g., operating systems, architecture, networks, databases, software construction and testing) and computational science. Paradoxically, experimental computer scientists have never felt completely welcome in the university. Many of them encounter difficulty with academic tenure processes, where the commonly-applied rules for peer recognition in mathematics and engineering science (counting publications) don’t carry over well for systems [snyd94]. At the same time, many of them find themselves attracted to industry by higher salaries and better laboratories, especially in times of high demand: the late 1970s were one such time and the late 1990s another.

Two excellent early examples of experimental work were virtual memory and performance analysis -- studies that led to the development and validation of useful, lasting theories and to practical systems [denn81perf]. Yet such successes have been the exception, not the rule. Marvin Zelkowitz and Dolores Wallace found that fewer than 20% of 600 papers advocating new software technologies offered any kind of credible experimental evidence in support of their claims [zelk98]. Walter Tichy is more pointed: he claims that many academic computer scientists have a lackadaisical attitude toward experimental work, which impairs its quality and novelty [tich98].

At the heart of this paradox are different, unreconciled views of programs and programming. Computing theorists are inclined to think of programming as a mathematical exercise, a process of guaranteeing that an algorithm meets its input-output specifications; yet formal methods seem capable of delivering only a small fraction of useful software systems in acceptable time. Engineers are inclined toward trial-and-error prototyping; yet many software systems are delivered late and over budget, with almost no analysis of their properties or performance. In reality, each approach offers benefits; finding a synergistic common ground has not been easy.

This paradox exacted a toll during the brain drain of the 1970s. In 1979 Jerome Feldman warned that experimental computer science was in jeopardy; he called for more competitive academic salaries and for explicit NSF support of experimental computer science [feld79]. The ACM Executive Committee endorsed the report while warning against equating “tinkering” with “scientific experimentation” [denn81ecs, denn79]. The chairs of the computer science departments soon echoed similar sentiments [denn81sno]. In 1989, the ACM/IEEE committee on the core of

computer science, which I chaired, reaffirmed that computer science gets its unique character from the interplay of theory, abstraction, and design [denn89]. (We used abstraction to refer to the scientific method, which includes modeling and experimentation.) It's like a stool -- remove any one of the three legs and it falls over.

Despite these encouragements from their leaders, many academic computer scientists continued to view experimentation as lower in status than theory or design. The National Research Council twice called our attention to this alarming drift, with limited success (See Hartmanis [hart92], and Snyder, 1994 [synd94]). We never fully recovered from the late-1970s brain drain. The stool continues to list.

Computational Science

Computational science is scientific investigation through modeling and simulation of physical processes on computers. Science is traditionally seen as a paradigm for discovering the laws of nature: the paradigm consists of forming a hypothesis, making predictions based on the hypothesis, collecting data, and analyzing the data for confirmation or denial of the hypothesis. Hypotheses are often formulated as mathematical models that can be used to calculate values of interest in the investigation. In science, theorists concentrate on formulating theories and mathematical models of physical processes. Experimenters concentrate on building instruments and using them to acquire data for subsequent analysis. Computation is now seen as a third approach: a model or simulation of the physical process can be measured without building a specialized instrument and transporting it to a difficult environment.

Most of those working in computational science say that progress comes partly from hardware and partly from software. In the first forty years of computing, computational speeds increased by about 10^6 from hardware improvements and 10^6 through software (algorithm) improvements -- a staggering 10^{12} combined improvement. These figures confirm that the goals of computational science can be realized only with close collaboration between computer scientists and physical scientists -- the former understand architectures and algorithms, the latter the physical processes and mathematical models in their disciplines.

The notion that computation is a third paradigm of science was accepted widely by the mid-1980s. It grew out of an impressive record of supercomputing successes in diverse fields such as aeronautics, astronomy, Bayesian inference, chemistry, combustion, cosmology, earthquake prediction, materials, neuroscience, oceanography, oil exploration, statistics, tomography, and weather forecasting. Leaders in these fields banded together and defined the next generation of problems in their areas as "grand challenges". They received a big impetus when Ken Wilson received a Nobel Prize for his computational physics work on magnetics; Wilson called for massive investment in parallel supercomputers that could run at billions and eventually trillions of operations per second. (The prevailing top speeds of supercomputers were hundreds of millions of operations per second.) These developments caught the attention of US Senator Albert Gore, who fought for and

won congressional passage of a national High Performance Computing and Communication Initiative (HPCCI), which was signed into law in 1989. Similar initiatives were started in Europe and Asia.

Most computer scientists stood at the sidelines while all this was happening. Within the discipline, the numerical analysts resonated with computational science. But many of their colleagues did not, seeing computing in science as “applications” of minor consequence to computer science. They practiced their beliefs: aside from numerical analysts, few computer scientists were involved in cross-disciplinary research teams. Among those who were, many found themselves paired with scientists who regarded them not as peers but as programmers. Wilson and others, claiming non-cooperation from computer scientists, proposed forming their own departments of computational science.

Fortunately for the discipline, such proposals did not result in a widespread movement to establish separate computational science departments and institutes. Instead, the large influx of research funds under high-performance computing initiatives enticed many computer scientists to join cross-disciplinary teams after all. Today, many computer science departments embrace computational science and collaborate with other science departments. The numerical analysts are now called computational scientists and have been integrated into the mainstream. The pragmatic interests of scientists in other fields have enriched the discipline.

Software Engineering

Recent proposals in several states to license software engineers have strained tensions between computer scientists and software engineers. Software engineers tend to believe that certification is valuable and licensing is inevitable; they want significant changes in the curriculum for professional software engineers. Other computer scientists tend to believe that certification is not a proper job for a university degree program and that licensing would be harmful because it would lock in minimal standards in a changing field of rising standards. Frustrated, a growing number of software engineers want to split off from computer science and form their own academic departments and degree programs. Noting other dualities such as chemical engineering and chemistry, they ask, why not software engineering and computer science? [parn97] [denn98]

No such rift existed in the 1940s and 1950s, when electrical engineers and mathematicians worked cheek by jowl to build the first computers. In those days, most of the mathematicians were concerned with correct execution of algorithms in scientific application domains. A few were concerned with models to define precisely the design principles and to forecast system behavior.

By the 1960s, the mathematicians had evolved into scientific programmers (who used languages such as Fortran, Algol, and Lisp). A new kind of programmer (who used Cobol and database languages) had been born of business applications. The engineers who built computers and these various breeds of programmers were ready for marriage, which they consummated and called computer science. But the

same tensions described earlier were present. The descendants of the original mathematicians and engineers instinctively sought respect from traditional scientists and engineers; they loathed a lack of rigor in application programming and feared a software crisis. Professional programmers found little in computer science to help them make practical software dependable and easy to use. Software engineers emerged in the late 1960s as the pragmatists, responding to the needs of professional programming by adapting computer science principles and engineering design practice to the construction of software systems.

Software engineers identified more with the engineering professions than with the sciences. They developed professional standards of ethical conduct. They paid a great deal of attention to design. (Terry Winograd, however, worries that they do not pay enough attention to the human side of design, and that an important new field, software architecture, may have to develop on its own [wino97].)

Opinions differ on whether the field has matured enough to permit the software engineers to follow a different path from computer science. Even if they do separate, they will both be part of the Profession of Computing and will share a common scientific core [denn89].

Basis of a Profession

The short history above depicts a young profession struggling to establish a permanent identity in a skeptical world seeking pragmatic returns. As the 1990s draw to a close, computers have infiltrated every aspect of business and life and there is no longer any doubt that computer science is here to stay. The real question is whether academic computer science will adapt to the demands for a profession.

A prerequisite for adaptation is a clear understanding at what our profession is and what it needs to become if it is to serve the hundreds of millions of people who depend on computers and networks. That understanding will be the basis of our approaches to education and research. It will suggest answers to such basic questions as: What are we preparing our students for? What concerns must our students learn to listen for and take care of? What must we investigate in our research labs?

Today, most computer scientists understand computer science as a discipline that studies the phenomena surrounding computers. These phenomena include design of computers and computational processes, representations of information objects and their transformations, theoretical and practical problems in hardware and software, efficiency, and machine intelligence. In Europe the discipline is called “informatics” and in the USA “the discipline of computing” or “information technology”. The computing profession is understood as the set of people who make their livelihood by working with information technologies.

This is the common-sense interpretation of the computing profession. I believe it is too narrow and, in its narrowness, it is misleading. I believe it is the source of the tensions discussed earlier and an impediment to the kind of profession sought by

the vast majority. The good news is, we can retrain our common sense. We can begin by examining other professions.

Underlying every profession is a durable domain of human concerns and breakdowns. Breakdowns are events that interrupt the expected flow of actions or work; these events may be the unanticipated failure of some person or system to deliver an expected result, or they may be the unexpected appearance of new challenges and opportunities. Durable means that the breakdowns and concerns are long-lasting, if not permanent: they are inevitable and they are recurrent. The profession is the set of people, institutions, and practices for taking care of people's recurrent breakdowns and concerns in the domain. Clients expect professionals to be ethical, responsible, and competent -- consequently, the profession includes institutions that declare and enforce standards of conduct, and institutions that train and certify competence. Three examples illustrate. Medicine addresses a permanent concern of all human beings, law a permanent concern of most, and libraries a durable concern of many:

- (1) Health is a permanent concern of all human beings. Breakdowns in health are inevitable because of disease, accident, or aging. Health care professionals take care of people's concerns and breakdowns in health. Hospitals, HMOs, insurance companies, government health programs, the national medical association, the medical "colleges," and medical schools are the principal institutions of this profession. Doctors must be licensed to practice medicine and can obtain certificates testifying to higher levels of competence in specialties. Doctors who violate professional standards are subject to reprimand or censure by the national medical associations, malpractice lawsuits, and loss of license.
- (2) The rule of law is a permanent concern of most human beings. Most people live in societies with governments, constitutions, legislatures, and laws. Implementing agreements and carrying out actions without violating laws or incurring penalties is an ongoing concern for them. Breakdowns are inevitable because people do break laws and because many business practices are governed by contracts. Two allied professions help people deal with their concerns and recurrent breakdowns about laws: the legal profession (lawyers, judges) and the law enforcement profession (police, other law enforcement agents). Law schools, police academies, legislatures, courts, and the national legal and police associations are the principal institutions of these professions. Lawyers must pass a bar examination and be licensed to practice law. Lawyers who violate professional standards are subject to reprimand or censure by the legal association, malpractice suits, and loss of license. Similarly, police are trained rigorously and are subject to sanctions.
- (3) The preservation of sharing of recorded human knowledge is a durable concern of many human beings. Progress in technology, law, commerce, politics, literature, and many other aspects of civilization depends on access to knowledge created by our ancestors. Civilizations can be interrupted or lost when they lose access to their own historical documents and records.

The profession of library science helps people deal with these concerns by preserving documents, making them available publicly, cataloging and organizing them, and preserving them. Libraries, schools of library science, and library associations are the principal institutions of this profession. Librarians must earn certain credentials to practice the profession and are subject to reprimand or censure by their professional associations.

To what extent does our computing profession address durable concerns and breakdowns? Demand and enforce standards of conduct? Certify competence of its members? Have analogous institutions?

The durability criterion is clearly met: computation and coordination of action are ongoing concerns and sources of breakdowns for all human beings. Let me explain. Ours is a world of information and numbers, mostly processed by machines and transmitted by networks. Telephone and fax are ubiquitous, the Internet soon will be, and databases are springing up like weeds everywhere in the Internet -- all technologies that extend the distance and time over which people can successfully coordinate actions. Nearly everyone in every developed country is affected by digital telecommunications; leaders in underdeveloped countries are aggressively installing informational infrastructures to accelerate their countries' entries into world markets. In the same way, computation is an integral part of the daily practices of finance, engineering, design, science, and technology. Word processing, accounting, databases, design automation, and report writing software impact every other profession. The digital world offers many new kinds of breakdowns, ranging from failures of computers and communications, to software bugs, to the challenge to install software that improves an organization's productivity. The computing profession is the set of people and institutions who take care of people's concerns in information processing, computation, and coordination over networks of computers.

These concerns are bigger than are implied by the phrase "phenomena surrounding computers". They include, as is commonly understood, the design and analysis of hardware and software to perform new functions or to perform old functions in new ways. But these concerns also include the design, installation, configuration, operation, and maintenance of reliable computer systems within homes and organizations. They include standards for communication and information exchange. They include privacy and integrity of conversations, files, and documents in networks of computers. They include working with the customer to design computer systems that support the work of the customer's organization. They include the shared values and glorious histories of the people in the profession and others who use computers and networks.

In other words, the concerns are not phenomena that surround computers. It is the other way around. The computers surround the concerns.

The language of "phenomena surrounding computers" increasingly exposes computer scientists to isolation from the concerns people have about information processing and communications. People turn to professionals for the help they

need. There will be a computing profession, but some of today's computer scientists will never learn to be part of it.

I am often asked, "Isn't the pursuit of clients' concerns incompatible with the need for basic research?" I see no incompatibility. The question assumes that client concerns are short-term and research long-term. It is a false dichotomy. I do see a lack of skill in articulating the connections between research questions and what people are concerned about. Medical researchers, for example, run plenty of esoteric, highly technical projects without an immediate payback. But they talk differently about their work. Listen to an example: "Even though sequencing the human genome is pretty technical, we believe we're hot on the trail of a cure for Alzheimer's disease." In contrast, the researcher who says, "The question I'm studying has been open for many years and I'm having fun trying to settle it," does not connect to a client's concerns. The latter response is about the speaker not the listener. It's not that such a researcher isn't working on something important; what's missing is the practice of articulating the connection with people's concerns.

What about the other aspect of profession, standards of conduct and competence? In this area we are even more immature than we are in listening to and acting on concerns. Our professional societies (ACM and IEEE mainly) have standards of conduct -- but do not enforce them. We have yet to develop criteria of competence and to ask our colleges and universities to certify their graduates.

Practices

Practices are habits, routines, processes, and skills performed by individuals and groups mostly from experience and with little thought [spin97]. Practices are a marvelous invention -- they enable us to get things done quickly, without reflection. Practices are "embodied" or "ready to hand" knowledge. Practices are learned by doing and by involvement with people who already embody them; they cannot be learned by "applying" mental or descriptive knowledge. Mental knowledge and practices are different forms of knowledge; the one does not imply the other. Yet practices are held in lower regard than mental knowledge by many academics, who value "reflective action" more than "reflexive action". Trying to understand knowledge without understanding practices is like expecting to play par golf after reading a book on the physics of golf swings modeled as pivoted pendulums.

It is impossible to discuss a profession without discussing practices. There are three reasons for this. First, professional competence is judged by observing a person's practices to determine whether the person is capable of fulfilling standard requests without intervention of a supervisor [drey92]. Second, ethical behavior is also a practice of conforming one's actions to preset community standards of right and wrong, integrity, and honesty. Third, professions are always concerned with innovations. Innovations are shifts of practices that enable the practitioners to be more productive in some way. Until an idea is practiced, it is no innovation. (More will be said about innovation shortly.)

Practices are not just personal. They exist in communities of people, where they manifest themselves not only as shared habits, routines, and processes, but also as a shared “common sense” of the community. The common sense informs people what is acceptable or not, what is true without proof or not, what fits or does not fit, and the like [spin97]. Many professional communities also set standards of performance and maintain institutions that certify competence at different levels. Certification is another name for the public demonstration of competence. In some cases, such as engineering, education, accounting, law, or medicine, certification can be quite specific and rigorous. In these cases, certificates are necessary or at least highly desirable for professional practice.

Within the university, there is a vigorous debate on whether practices should be accorded greater importance in higher education. This debate has been triggered by the recurrent call for competence. Students and employers ask for educational programs that confer and then certify definite skills. Given that so many people now view a college diploma as a ticket to a good job, and that so many employers recruit directly from universities, this is no surprise. Yet this call inspires derision from some faculty, who hear the “competence” as a code word for vocational “training” and who argue strenuously that it is not the mission of a university to provide training. They view courses aimed at skills as steps in the direction of increasing specialization, an affront to the university’s mission of general education.

Other educators argue just as strenuously for more proficiency-based courses, which means that students don’t pass until they can *demonstrate* that they know the material and can act effectively with it. To reassure their colleagues, these educators say they mean competence in a broad sense that ranges from operating a computer or building a large software system to public speaking, rhetoric and debate, critical thinking, analyzing history, working on and managing teams, and leading a group. Certification is another name for the public demonstration of competence. In some cases, such as engineering, education, accounting, law or medicine, certification can be quite specific and rigorous. Certificates are necessary or at least highly desirable for professional practice.

This debate is the first sign of an important change in our understandings of data, information, knowledge, and practice. It is seeping into more people’s consciousness that there are fundamental distinctions among these four, which may be described as follows. (1) Data are symbols inscribed in specified patterns by human hands or by instruments. (2) Information is the judgment, by an individual or group, that given data resolve questions, disclose or reveal distinctions, or enable new action. In other words, information is data that makes a difference to someone. Information thus exists in the eyes of the beholder; the same data can be nonsense to one person and gold to another. (3) Knowledge is the capacity for effective action in a domain of human practice. (4) Practices are recurrent patterns of action that effectively accomplish certain objectives with little or no thought. Practices are a form of embodied knowledge.

Lewis Perelman likens these distinctions to eating in a restaurant. The data are the symbols on the menu; information is the understanding of what the menu offers;

knowledge is the dinner; practice is the digestion that turns the dinner into useful nutrients [per192].

These distinctions are not practiced rigorously in the university. Most curricula are set up on the assumption that there is a body of knowledge (organized data about a field that conveys information to its beholders) that must be transmitted to the students. The teacher is the communication channel. Testing reveals whether the information survived transit intact. Universities are serving mostly menus. The call for competence is a cry from the hungry for nourishment.

The growing awareness of these distinctions will engender significant shifts in education. The student-teacher relation of “apprentice-master” will become a more traveled path to knowledge. The teacher will need special skills, not at presenting information, but at observing and shifting how students see and bring forth their worlds [schn98]. The apparent contradiction between general and professional education will disappear. General education seeks to produce a graduate who can act effectively by reading, writing, speaking, and listening, and who understands history, literature, philosophy, language, and social relationships. General education is the context in which a person can attain higher levels of professional competence.

Applications

In most professions, the word “application” is used to distinguish theory from practice: practice appears not as a form of knowledge, but as application of theory. In the computing profession, this meaning is specialized to denote programs that perform tasks for non-programming users in particular domains; application programs apply the results of theory to the practices in which the users are engaged. Scientific applications include statistical analyzers, equation solvers, chemical bond analyzers, ground soil diffusion analyzers, and fluid flow solvers. Medical applications are programs such as patient record managers, EKG analyzers, and expert systems for diagnosis and prescriptions. Commercial applications include graph generators, word processors, spreadsheets, database systems, accounting and payroll systems, report generators, and programming environments. Each domain of practice has its own list of programs of this kind.

Computer science researchers also use the term “application” in a much narrower sense. They use it to distinguish questions of immediate and transient concern to practitioners from research questions of lasting significance. Many computer scientists see “applications” as the inverse of “research”; time spent on applications is time not spent on research and does not earn a reward by the standards of scientific investigation. On the other hand, many business people see “applications” as their principal offer in the marketplace; they want computer scientists to collaborate with them in designing applications and they say they cannot otherwise “sell” research.

From the perspective of computing as a profession, research has a much broader role: research is a blend of “basic” and “applied”. Both serve the profession in their own ways, and the interaction between them strengthens the profession.

Innovation

Dennis Tsichritzis, the Chairman of GMD, the German National Research Center for Information Technology, argues that innovation is the ultimate objective of research [tsic97]. The sign of an innovation is new practices adopted by people in a domain, enabling them to be more productive at what they do. Inventions and good ideas are not innovations if no one uses them. There are at least four major processes of innovation, each supported by its own kind of research:

- (1) **Generating new ideas.** Powerful new ideas shift the discourse, in turn shifting the actions of those practicing the discourse. Research consists of formulating and validating the new ideas. It places a great deal of emphasis on originality and novelty. The scientific publication process aims to certify originality and novelty through peer review.
- (2) **Generating new practices.** A teacher or trainer inculcates people directly into the practices of a new discourse. Research consists of selecting, clarifying and integrating the principles relevant to the practices. It places a great deal of emphasis on understanding that produces competence.
- (3) **Generating new products.** New tools enable new practices; the most successful are those that enable people to produce their own innovations in their own environments. Research consists of evaluating and testing alternative ways of building a tool or defining its function. It places a great deal of emphasis on economic advantage.
- (4) **Generating new business.** Successful firms continually improve their business designs. Research consists of testing markets, listening to customers, fostering off-beat projects that explore notions defying the conventional wisdom, and developing new narratives about people’s roles and identities in the world. It places a great deal of emphasis on market identity, position, and exploring marginal practices.

Tsichritzis explicitly advocates the first three processes as the substance of a research center [tsic97]. Slywotzky advocates the fourth [slyw95]. Tsichritzis clearly practices the fourth his leadership of GMD.

Traditional computer science places the most value on the first of these four processes. The Profession of Computing will treat them equally.

Boundaries

Let us return to the subject of the boundaries of a field and its growth. Computer science itself originated at the boundaries between electronics, science, and the

mathematics of logic and calculation. During the early years (1950s through mid 1960s) the core areas of the discipline were numerical analysis, switching theory, logic design, and models of computation. Operating systems, compilers, databases, networks, and hardware processors were seen as applications. Computer scientists working at the boundaries with programmers of these applications discovered significant principles, which they incorporated successfully into proposals to include operating systems, compilers, databases, computer architecture, parallel systems, and distributed systems within the core.

It would be a mistake to think we have run out of new boundaries that have the potential to change the field. Look at a few of today's boundaries:

- New computing paradigms with biology and physics including DNA, analog silicon, nanodevices, organic devices, and quantum devices.
- Internet computations mobilizing hundreds of thousands of computers.
- Neuroscience, cognitive science, psychology, and brain models.
- Large scale computational models for cosmic structure, ocean movements, global climate, long-range weather, materials properties, flying aircraft, structural analysis, and economics.
- New theories of physical phenomena generated by “mining” patterns from very large (multiple) data sets.
- New approaches to storing, cataloging, locating, retrieving, and accessing documents and protecting intellectual property in the form of digital objects in the Internet.
- Workflow and coordination technologies from the business workplace, where improving productivity is a constant concern.

These boundaries are the likely sources of radical innovations. They are likely to yield new standard practices and core principles for computing in the next decade or two. Those who work the boundaries supply a life-stream that keeps the field vital.

The phenomenon of field boundaries is much deeper and is linked to entrepreneurship and the dynamics of professions [spin97]. Recall that professions form to take care of recurring breakdowns. A major breakdown's existence entices entrepreneurs to seek solutions. Entrepreneurs often find the seeds of solutions in anomalous practices that do not resonate with the current common sense of the field. The practices eyed by the entrepreneur may be central in another field. They must somehow be appropriated and adapted for the entrepreneur's field.

A short story will help clarify these statements. Early in the 1980s researchers in high-energy physics established bulletin board services to exchange preprints of physics papers. Within a few years they expanded their practice by storing physics papers on many servers in several countries. This created a breakdown for readers who wanted to see copies of cited papers: they had to open an FTP connection to the server containing the paper, transfer a copy, close the connection, and read the file

with a local word processor -- not exactly convenient. In the late 1980s, Tim Berners-Lee of CERN (Switzerland) invented a way to resolve this breakdown. He built the hypertext transfer protocol (HTTP), which would automatically fetch a remote paper when a reader mouse-clicked on a citation. The protocol wasn't user friendly -- authors had to learn a "hypertext markup language" (HTML) and write their papers in it. But it was good enough for the physicists because they could exchange their scientific findings much more rapidly once they learned the new language. Berners-Lee and his colleagues called their network of hyperlinked documents the World Wide Web [bern96int, bern96ppf].

In the early 1990s, Marc Andreessen of the National Center for Supercomputing Applications (NCSA) at the University of Illinois had been puzzling over a similar breakdown about sharing in the Internet [hafn96]. He invented the Mosaic Browser, a graphical interface that made it easy to view documents stored in the HTML format and to highlight links for easy mouse-clicking. With the browser, he was able to appropriate a practice from physics research into the mainstream Internet. He founded a company that eventually became Netscape. The browser revolutionized the Internet, transforming it into a household word and placing "http://" addresses on every business card and advertisement. Andreessen was an entrepreneur who transformed an anomalous practice into a central one. The breakdown that motivated him was resolved.

It is no accident that Andreessen's invention happened at the NCSA. Larry Smarr, the Center's director, himself a physicist, had dedicated the center to promoting interactions among disciplines. His project teams normally included computer scientists, physical scientists, and graphics artists -- the computer scientists worried about algorithm design and correctness, the physical scientists about the models and relevance to their discipline, and the graphics artists about the pictures for visualizing the massive data sets generated by the supercomputer. Smarr's practice of fostering interactions at the boundaries of current disciplines produced numerous scientific breakthroughs. The World Wide Web browser was one of the most prominent. (Andy Grove uses similar practices to foster innovation at Intel [grov96].)

The story does not end with Netscape's success. A profession has grown up around the World Wide Web. All the major builders of operating systems now seek seamless interfaces with the World Wide Web. Individuals and companies seek to project their personal and professional identities through web pages, web sites, and web services. In mid 1998 there were an estimated 80 million persons using the Web from 30 million computers offering well over 300 million web pages. With such a customer base, the long-floundering practices of electronic commerce took off as companies found successful business models for the Web; a growing number of companies did business only via their web sites. (The Amazon.com bookstore became a brand name and a model for other Internet businesses.) New jobs such as web master and web identity designer have appeared; none of these jobs existed in the early 1990s. Internet Service Provision (ISP) has become a booming business. The World Wide Web consortium (chaired by Berners-Lee) sets standards and charters improvements in protocols and markup languages.

Let me restate this in our terminology of professions. The Web profession exists to take care of people's concerns about projecting and protecting their identities in the web, about conducting business in the web, and about avoiding breakdowns such as broken connectivity, theft and fraud, and inability to communicate across boundaries. The Web was a radical innovation in communicative practices started by entrepreneurs who appropriated practices from physics researchers at a boundary with computer science.

Any profession that becomes insular will lose its access to the boundaries and with it the life-giving supply of innovations. The profession must value its boundaries and learn from its customers. Because information, communication, and coordination are fundamental human activities, computer science is likely to be involved with many fields and therefore to have many boundaries. Computer science, perhaps more than any other science, cannot avoid interactions with diverse groups of people.

It is even more important today than in the past to keep open the lines of communication among computer scientists, software engineers, and applications practitioners. Despite many differences, they can work together from a common interest in innovation, progress, and solution of major problems.

Disappearing Dichotomies

The framework for a profession of computing, sketched above, resolves four dichotomies that computer scientists struggle with today.

- (1) **Computer Science v. X**, for X being traditional computer science, information systems, information science, software engineering, computer engineering, database engineering, network engineering, systems engineering, software architecture, human-computer interface design, computational science, computational statistics, numerical modeling, and possibly one or two others. All these current disciplines are brothers and sisters in the family (profession) of computing. They have the same intellectual core, but different practices.
- (2) **Research v. Application**. Understanding research as generating new ideas is too narrow for the profession, which includes the other three other processes of innovation -- generating competence, generating products, and generating new businesses. Much innovation flows from the boundaries, where the current short-term concerns interact with long-standing professional practice. What is today called "application" is part of a continuum of research drivers within the profession of computing.
- (3) **Researcher v. Practitioner**. Some professional societies concerned with specialties of the Profession of Computing (e.g., ACM, IEEE, AAAI, SIAM) have a tendency to categorize people as "researchers", "practitioners", or "users" when defining their clients. These designations rankle many pragmatists, who do not themselves practice any of the computational arts

or sciences, or directly operate computational devices, but nonetheless depend on these technologies and have concerns about them. (For example, the many people interested in understanding and resolving the Y2K problem have found little help from any professional society.) Researchers, inventors, practitioners, users, pragmatists, and users -- all will be recognized as part of the profession of computing.

- (4) **Education v. Training.** Learning the professional practices of a specialty of information technology is every bit as important as learning the intellectual core of computing. The mark of a well-educated professional will be a balance of the two, earned perhaps through partnerships between universities and training companies. The current academic inclination to disdain skill-specific training does not fit a profession.

A Profession of Computing

In discussing the basis of any profession, practices, applications, and boundaries, I intended to ground these claims:

- (1) Most of those who use computers and communications do so through hardware, software, and networks whose inner workings are mysteries to them.
- (2) People in business and their clients, people at home, people in science and technology, and people depending on large software systems have concerns about the design and operation of reliable hardware, software, and network systems to help them do their work.
- (3) These people seek professional help in taking care of their concerns. They expect computing professionals to be responsive, competent, ethical, and able to anticipate future breakdowns.
- (4) The Profession of Computing is coming into existence to provide that help.
- (5) The education of computing professionals must account for practices as well as descriptive knowledge. It must include training as well as general education. It may not reside in any single university department, being distributed among computer science, software engineering, computational science, computer engineering, and related departments such as astronomy, physics, chemistry, biology, management science, linguistics, or psychology -- each of which contributes important specialties to the profession.
- (6) Individual computing professionals should embrace boundaries between their specialties and others in the profession. As a whole, the computing profession must embrace its boundaries with other fields to assure a constant stream of life-giving innovations.

Through its research, the Profession of Computing must anticipate future breakdowns that others will encounter. A close interaction between computer researchers and others is essential so that the questions under investigation remain

connected to real concerns, both short and long term. Otherwise computing research can drift into irrelevance and cease to earn public support.

Computer scientists and software engineers, who are at the heart of the computing profession, are being invited to embrace commercial applications, interactions with other fields, and the concerns of their customers. If they do not, clients of the profession will turn elsewhere for the help they need. It hardly needs pointing out that, in this case, computer scientists who do not do this will effectively isolate themselves from the Profession of Computing. An historical tendency toward insularity is, in my view, behind the current tensions between software engineers and other computer scientists.

This tension is, in fact, part of my motivation for writing this essay. I have been troubled during recent years by the skirmishing between software engineers and computer scientists, by the insularity of many computer scientists, and by the question of coping (in education) with the large demand from pragmatists for help. Somehow we have to adapt, take leadership, but give up our traditional feeling of “control” over the shape of the discipline. My conclusion is that we need to think in terms of profession rather than discipline, for there appear to be many disciplines that want to be part of the profession. That led me to enumerate everything that is involved in being a profession.

The academic entity most likely to succeed for the Profession of Computing is the College of Computing or the School of Information Technology headed by its own dean. This organizational unit would accommodate a significant subset and range of the specialties making up the profession -- which include traditional computer science, information systems, library science, information science, software engineering, computer engineering, database engineering, network engineering, systems engineering, software architecture, human-computer interface design, computational science, computational statistics, and numerical modeling. It would offer a common intellectual core and training in the practices of each specialty. It would offer certifications at several levels of professional competence in each specialty and would be dedicated to the ongoing support of the education needs of professionals. Its research programs would balance among the four major processes of innovation.

What of the questions about separation or reconciliation that vex traditional computer scientists and software engineers? Within the view of the Profession of Computing, the software engineers are part of the profession even they are not parts of traditional CS departments. Both groups have to come to grips with the fact that they are no longer in control of the profession; the pragmatists are. A bigger threat to the profession is a potential conflict at the dean’s level. If two deans divide the specialties between their schools without arranging for a common core and student interchange, there may be turf battles that will isolate the specialties and reduce communication among them, thereby weakening the Profession of Computing on that campus.

Computer scientists, software engineers, computational scientists, and other information technologists have a marvelous opportunity to transform their

academic disciplines into the Profession of Computing. They will have to face, and cross, the chasm between their practices as inventors and visionaries, and the pragmatic interests of their many clients and customers. It will not be easy. They have shown they can do it before, and they can do it again.

References

- [bern96int] Berners-Lee, T. "The Web Maestro: An interview with Tim Berners-Lee." *Technology Review* (July 1996).
- [bern96ppf] Berners-Lee, T. "WWW: Past, Present, and Future." *IEEE Computer* 29, 10 (October 1996), 69-77.
- [denn79] Denning, P., D. Brandin, and D. McCracken. "An ACM Executive Committee position on the crisis in experimental computer science." *ACM Communications* (September 1979).
- [denn80] Denning, P. "What is experimental computer science?" *ACM Communications* (October 1980).
- [denn81ecs] Denning, P. "Performance Analysis: Experimental Computer Science at its best." *ACM Communications* (November 1981).
- [denn81sno] Denning, P., et al. "A discipline in crisis -- the Snowbird Report." *ACM Communications* (June 1981).
- [denn89] Denning, Peter, et al. "Computing as a discipline." *ACM Communications* (January 1989). *IEEE Computer* (February 1989).
- [denn91] Denning, P. "Computing, Applications, and Computational Science," *ACM Communications* 34, 10 (October 1991), 129-131.
- [denn97] Denning, P. "How we will learn." In *Beyond Calculation: The Next 50 Years of Computing* (P. Denning and R. Metcalfe, eds), Copernicus (1997).
- [denn98] Denning, P. "Computer Science and Software Engineering: Filing for Divorce?" *ACM Communications* 41 (July 1998).
- [drey92] Dreyfus, H. *What Computers Still Can't Do*. MIT Press (1992).
- [feld79] Feldman, J., et al. "Rejuvenating experimental computer science -- A report to the National Science Foundation and others." *ACM Communications* (September 1979).
- [grov96] Grove, A. S. *Only the Paranoid Survive*. Currency Doubleday (1996).
- [hafn96] Hafner, K., and M. Lyons. *Where Wizards Stay Up Late: The Origins of the Internet*. Simon and Schuster (1996).

- [hart92] Hartmanis, J., et al. *Computing the future*. National Academy Press (1992). (<http://www.nas.edu>)
- [moor91] Moore, G. *Crossing the Chasm*. Harvard Business (1991).
- [parn97] Parnas, D. "Software Engineering: An unconsummated marriage." Inside RISKS column, *ACM Communications* 40, (September 1997), 128.
- [perl92] Perelman, L. *School's Out*. Avon (1992).
- [schn98] Schneiderman, B. "Related-Create-Donate: An educational philosophy for the cyber-generation." *Computers & Education* 31, 1 (1998), 25-39.
- [slyw95] Slywotzky, A. *Value Migration*. Harvard Business School Press (1995).
- [snyd94] Snyder, L., et al. *Academic careers for experimental computer Scientists* (1994). National Academy Press. (<http://www.nas.edu>)
- [spin97] Spinoza, C., F. Flores, and H. Dreyfus, *Disclosing New Worlds*, MIT Press (1997).
- [tach98] Tachibana, Takashi. "Closing the knowledge gap between the scientist and nonscientist," *Science* 281 (7 Aug 1998), 778-779.
- [tich98] Tichy, Walter. "Should Computer Scientists Experiment More?" *IEEE Computer* (May 1998).
- [tsic97] Tsichritzis, D. "The dynamics of innovation." In *Beyond Calculation: The Next 50 Years of Computing* (P. Denning and R. Metcalfe, eds). Copernicus Books (1997).
- [wino97] Winograd, T. "Interaction Design." In *Beyond Calculation: The Next 50 Years of Computing* (P. Denning and R. Metcalfe, eds). Copernicus (1997).
- [zelk98] Zelkowitz, M., and D. Wallace. "Experimental models for validating technology." *IEEE Computer* (May 1998).