Chapter 23

When IT Becomes a Profession

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

5 / 30 / 01

This essay has been prepared for the ACM1 book, THE INVISIBLE FUTURE, to be published by McGraw Hill in August, 2001.

Peter Denning is Past President of ACM, past director of the ACM Information Technology Profession Initiative, inventor of the working set model for program behavior, and a pioneer of virtual memory systems and of computer system performance theory.

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 Information Technology professionals to help them with their needs for designing, locating, retrieving, using, configuring, programming, maintaining, and understanding computers, networks, applications, and digital objects. Students expect Information Technology curricula to provide comprehensive coverage of all technical, research, and leadership principles and practices needed to make them effective professionals; they rely especially on the faculty for a comprehensive view of a fast-changing, fragmented world, for assistance in framing and answering important questions, and for training in standard professional practices. Professionals expect their professional societies to support their identities as professionals, to advocate lifelong continuing professional education, and to speak out in public policy issues affecting Information Technology. In short, everyone has greater expectations of IT professionals than of the Information Technologies themselves.

But the reality of what users, students, and professionals find differs markedly from what they expect. They find poorly designed software, complex and confusing systems, crash-prone systems, software without warranties, begrudging technical support, surly customer service, inter-vendor finger-pointing, disregard for privacy, and even poorly-managed, investment-squandering dot-com companies. Business people find it difficult to find qualified IT workers and then keep them current with a fast-changing body of knowledge. Students find IT curricula that focus more on programming than on systems, on theory more than experimentation, and on concepts more than on practice. Professionals find little help for lifelong learning or career advancement and a cacophony of conflicting voices from professional groups. Users -- by far the largest group -- are growing increasingly intolerant of these problems (Dertouzos 2001). They expect IT professionals to organize themselves more effectively in order to address the problems and serve their customers. Why is this not happening?

There are today over forty organized professional groups in computing and information technology. (See Table 1.) The IT-specific disciplines are the core technologies of computer science and engineering; the people working in these disciplines are called *computing technologists*. The IT-intensive disciplines are the other branches of science, engineering, and commerce that innovate in IT as part of their work. The IT-supportive occupations are relatively new professional specialties that support and maintain IT infrastructure**.** These groups all share a common scientific core in IT but have different professional practices and concerns. Taken together, these groups constitute the emerging Profession of Information Technology.

IT-Specific Disciplines	**IT-Intensive Disciplines**	**IT-Supportive Occupations**
Artificial intelligence	Aerospace engineering	Computer technician
Computer science	Banking and financial services	Help desk technician
Computer engineering	Bioinformatics	Network technician
Computational science	Cognitive science	Professional IT trainer
Database engineering	Digital library science	Security specialist
Computer graphics	E-commerce	System administrator
Human computer interaction	Genetic engineering	Web services designer
Network engineering	Information science	Web identity designer
Operating systems	Information systems	Database administrator
Performance engineering	Public Policy and Privacy	
Robotics	Instructional design	
Scientific computing	Knowledge engineering	
Software architecture	Management information systems	
Software engineering	Multimedia design	
System security	Transportation systems	
	Telecommunications	

Table 1: Subdivisions of the IT field

Traditional computer scientists face a dilemma. Should they insist that newcomers join and that their offspring not separate? If so, they run the risk of being sidelined in the new profession. Should they cross the 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 and a new view of their customers.

The purpose of this essay is to paint a picture of an IT profession -- its knowledge, its core values, and its standard practices -- and the breakdowns blocking it. This will map an agenda for industry, academia, and professional societies to help today's computing technologists become true professionals.

Four Foundation Words

Ours is a field of buzz words whose meanings have blurred under a barrage of flashy vendor advertisements. Nowhere is the blur more obvious than with four words at the very foundation of our profession. The distinction between "data" and "information", once carefully observed by computing professionals, has all but disappeared. "Knowledge" has been trivialized to the content of databases. "Practices" are no longer seen as an important form of knowledge. Our sloppiness with these terms undermines our credibility with others, who wonder whether to believe our claims for a solid scientific basis, for effective professional education, for productivity-enhancing business systems, or for safe and dependable software. In the remainder of this essay, these four foundation words are defined so:

- Data are symbols inscribed in formalized patterns by human hands or by instruments.
- 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 the meaning that someone assigns to data. Information thus exists in the eyes of the beholder; the same data can be nonsense to one person and gold to another.
- Knowledge is the capacity for effective action in a domain of human practice.
- Practices are recurrent patterns of action that effectively accomplish certain objectives with little or no thought. Practices are embodied knowledge.

Lewis Perelman (1992) 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.

Although these distinctions are not practiced rigorously in the university, their widening adoption will almost certainly engender significant shifts in education. The student-teacher relation of "apprentice-master" will become a frequently traveled path to knowledge. The teacher will need to be competent both as a presenter and as a coach (Schneiderman 1998).

Basis of an IT Profession

Today, most people understand computer science as a discipline that studies the phenomena surrounding computers (Denning 1989). These phenomena include design of computers and computational processes, representations of information objects and their transformations, hardware, software, efficiency, and machine intelligence. In Europe the discipline is called "informatics" and in the USA "the discipline of computing." The computing profession is understood as the set of people who make their livelihood by working with computing technologies.

But "making a livelihood" is a narrow view of profession. Five hallmarks are readily visible in other professions:

- A durable domain of human concerns.
- A codified body of principles (conceptual knowledge).
- A codified body of practices (embodied knowledge)
- Standards for performance.

- Standards for ethics and responsibility.

The durability criterion is the most fundamental of the five. It means that there is an ongoing, universal set of recurrent breakdowns and concerns affecting large majorities of people. It makes the formation of a profession an historical necessity.

When there is an enduring domain of concerns, professionals are needed to take care of recurring breakdowns and to help people realize opportunities with complex technologies. Professionals operate from codified bodies of principles and practices, which, as we have seen, are distinct and equally important forms of knowledge. The standards of performance are essential to inspire trust in the methods of the profession and in the work of individual professionals. Because of the ubiquity of the concern and the breakdowns surrounding it, and because of the risks to life and business from poor practice, professions find themselves subject to strong social pressures for competent, ethical practice. A profession includes institutions for preserving the knowledge and the practices, defining and enforcing the standards, and educating professionals. The professions of medicine, law, libraries, and management prominently exemplify these principles. (Denning 1991)

How does IT measure up by these criteria?

(1) Durability. This criterion is clearly met: effective communication and, to a lesser extent, computation are ongoing concerns and sources of breakdowns for all human beings. Ours is a world of information and numbers, many 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 and participate in communities. Nearly everyone in every developed country is affected by digital telecommunications; leaders in many underdeveloped countries are aggressively installing informational infrastructures to accelerate their countries' entries into world markets. 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 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 measurably improves an organization's productivity.

(2) Body of Principles. This criterion is clearly met. Our conceptual knowledge is codified in the curricula of our degree and training programs. Professional societies maintain curriculum guidelines for computer science, computer engineering, and software engineering degree programs.

(3) Body of Practices. This criterion is not met. Few university programs define standard practices in various IT specialties and make sure their students learn them. Professional societies in IT offer no guidelines. Software engineers, who are well ahead of other specialties in defining their standard practices, have an informal list that includes abstraction, setting specifications, object-oriented design, reuse, and testing; but no curriculum guideline insists that students learn them (Meyer 2001). The growing interest among academic leaders to form IT colleges is prompting a new look at professional practices in the curriculum.

(4) Standards of Performance. This criterion is not met. Few customers of IT know what to claims they can trust about software design and implementation. Few university programs define criteria for different levels of professional competence or offer certification tests. Almost alone among professional organizations, the British Computer Society has taken a leading position in defining IT competencies. The Institute for Certification of Computer Professionals (ICCP) does this in a narrow area but is not widely known or used. The IEEE Computer Society has a Certified Software

Engineering Professional program. Individual states are showing an interest in licensing software engineers.

(5) Ethics and Responsibility. The professional responsibility criterion (ethics and standard practice) is partially met. The professional societies (ACM and IEEE, for example) have codes of ethics but do not enforce them. There are all too many discontent users, a signal that we are not listening carefully to them and serving them.

It is useful to distinguish crafts, trades, and disciplines from a profession (Holmes 2000). A craft is a set of practices shared by a community of practitioners but does not enjoy a recognized social status. A trade is an organized group of practitioners (some may be craftsmen) subject to standards imposed by government in return for freedom to practice the trade. A discipline is a well-defined field of study and practice. A profession includes related disciplines, trades and crafts. It embodies a core value of listening to its clients and for being socially responsible.

The US Department of Education, acting under a congressional mandate, has defined a profession as a set of people who have at least two years of post baccalaureate education and whose field is on an approved list. This definition is much less rigorous than ours.

So we are in paradox. The IT profession is an historical necessity and yet the IT field has progressed little beyond being a collection of crafts. What blocks progress? The main impediments are in the practices and performance areas -- the very areas in which the customers of IT are most important. The problem is that IT's way of looking at itself is lopsided toward the technology and is therefore self-limiting. Approaching the design of software and services with customers at the center runs against the grain of our field. We need a major shift of world-view to cross that chasm.

Some business people do not yet see the profession as an historical necessity. They think that professions tend to regulate who can be a member and to impose training and competency requirements. They can be forgiven for not welcoming restrictions on who can be an IT professional when the market demand for IT professionals outstrips the capacity of schools to train new professionals. But this is a short-term issue. In the long term, the forces compelling profession will prevail.

In the meantime, the proliferating collection of IT specialties has not begun to coalesce into a clearly defined and coherent profession that practitioners can identify with. This lack of unity has made it difficult for the IT field to have its own voice and to address problems that affect the entire field -- notably unreliable and unsafe software systems, chronic shortages of IT workers, shortages of faculty and graduate students, and fragmentation among the professional societies.

Customers Across the Chasm

It is an irony that the computing discipline, which gave birth to the IT profession, is no longer the driving force in the profession. The field is being driven by the large numbers of users with mundane, practical concerns about using and relying on computers; and by many powerful business, civic, government, and industry leaders. Computing technologists are the inventors and visionaries. They are admired for independence, entrepreneurship, invention, and vision. Computing technologists, however, need to come to grips with the fact that they are no longer in control of the field. They are no longer the primary inventors of hardware and software. Their research is no longer the primary impetus behind most IT innovations. They are one among many professional groups in the field. Why is this?

I believe that computing technologists are experiencing a phenomenon described eloquently by Geoffrey Moore in 1991. No relation to Gordon Moore (the Intel founder famous for the 18-month doubling law of processor power), Geoffrey Moore was a principal of the Regis McKenna advertising agency headquartered in Silicon Valley. Well before the "dot-com boom," Moore had witnessed hundreds of new technology 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 unexpectedly 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 laggards, and the ultra-conservatives. 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 70-80 percent of the population, 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 exhaust the small market of visionaries and attempt 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; they want to trust their suppliers and the professionals who help them. 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 technologists and other IT specialists are the inventors and visionaries in Moore's model. The multitudes of users are pragmatists, whose concerns and demands differ sharply from those of early-adopters. Computing technologists 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 multitude of pragmatists.

Putting it more bluntly, we computing technologists do not understand that we have customers. Customers are not abstract entities with Internet accounts who buy computers. They are people with concerns, breakdowns, hopes, fears, and ambitions. Some of us understand this as individuals but collectively we do not. We do not see that our success depends on bringing value to our customers. Many of us do not know what "bringing value" means. The language we use to describe our field -- "phenomena surrounding computers" -- focuses on the technology and pushes our customers into the background. The phrase "information technology" suffers from the same problem. We pursue research questions to satisfy our own curiosity but build few stories of how our results will benefit our users -- and then we wonder why research funds are in short supply. Our budding entrepreneurs start companies based on "cool technologies" -- and later wonder why venture funds are hard to come by or why their companies go belly-up. Computing faculty teach the technology but not the leadership, management, and people skills professionals need. We have allowed ourselves to remain isolated 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 many of us will never learn to be part of it.

The chasm between scientists and citizens who live and work with technology extends much further than computing. Science journalist Takashi Tachibana (1998) says that the chasm between technologists and non-technologists widened during the 20th Century into a gulf. Unless technologists can find ways to communicate effectively with the masses, the basic research enterprise feeding technological development will dry up and the average person will be unable to make well-grounded assessments about technology.

Struggles in the Growth of Computing

Moore's model suggests that a successful organization must chart a growth process that gradually expands to larger markets of people with different mindsets. 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. The earlier crossings were the marriage of the separate roots of mathematics, electrical engineering, and science into the single discipline of computer science (1960s), embracing systems into the core of computing (1970s), embracing computational science (1980s), and embracing various branches of engineering such as software, computer, database, network, graphics, and workflow (1990s).

An Unexpected Marriage (1960s)

Computer science boasts strong historical roots in engineering, mathematics, and science. The science roots, exemplified by 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, exemplified by DaVinci, reflect interests to harness the laws of nature through construction of artifacts and systems; in the 20th century, electrical and electronic systems were central. 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, Church, and Turing in the 20th.

Scientists, engineers, and mathematicians 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.

In 1951 the first commercial computer (Univac) was delivered. Since that time, the demand for people trained in building hardware, programming software, and applying computing in various disciplines has stimulated formal university degrees in computing. The first degree program in computing was offered at the University of Pennsylvania around 1960 and the first computer science departments were formed at Purdue and Stanford in 1962. The ACM (Association for Computing Machinery) produced its first formal computer science curriculum recommendations in 1968.

By the early 1970s, the several dozen computer science departments found themselves under increasing pressure from industry to include systems specialties in their conception of the required core courses. The old core subjects -- favored by the inventors -- were numerical analysis, automata theory, language theory, switching theory, combinatorial and discrete math, and mathematical logic. The new core subjects -- favored by the pragmatists -- were architecture, operating systems, compilers, databases, networks, software systems, and graphics systems. By 1975, the inventors and pragmatists, working together, had designed curricula encompassing both the theory and practice of computing. This chasm-crossing has had enormous consequences. By 2000, there were nearly 200 PhD-granting computer science departments in the US and Canada, and over 1500 other colleges offering computer science degrees.

Experimental Computer Science (1970s)

Experimental methods are at the heart of the systems areas (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 common criteria for peer recognition in mathematics and engineering science (counting publications) do not carry over well for systems (Snyder 1994). 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 and early 2000s 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 (Denning 1981a). Yet such successes have been the exception, not the rule. Marvin Zelkowitz and Dolores Wallace (1998) found that fewer than 20% of 600 papers advocating new software technologies offered any credible experimental evidence in support of their claims. Walter Tichy (1998) is more pointed: he claims that many academic computer scientists have a lackadaisical attitude toward experimental work, which impairs its quality and novelty.

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. 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. The ACM Executive Committee endorsed the report while warning against equating "tinkering" with scientific experimentation (Denning 1979, 1980). The chairs of the computer science departments echoed similar sentiments (Denning 1981b). In 1989, the ACM and IEEE reaffirmed that the unique character of computer science flows from the interplay of theory, scientific method, and design (Denning 1989). It's like a stool -- remove any one of the three legs and it falls over.

Since crossing this chasm, academic computer scientists have needed constant encouragement to view experimentation as equal in status than theory or design. The National Research Council twice called our attention to it; see Hartmanis (1992) and Snyder (1994).

Computational Science (1980s)

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 process consists of forming a hypothesis, making predictions based on the hypothesis, collecting data, and analyzing them 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. Computation enables modeling or simulation of the physical process without building a specialized instrument and without closed-form mathematics.

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 106 from hardware improvements and 106 through software (algorithm) improvements -- a staggering 1012 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 work in physics; 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. This marked the successful crossing of the chasm separating computer scientists from the rest of science.

Within computer science, the numerical analysts resonated most with computational science. Aside from them, few computer scientists were involved in cross-disciplinary research teams. Those who were involved found themselves teamed 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 much action. Instead, the large influx of research funds under high-performance computing initiatives enticed many computer scientists to join cross-disciplinary teams. Today, many computer science departments embrace computational science and collaborate with other science departments. The numerical analysts are now called computational scientists. The pragmatic interests of scientists in other fields have enriched the discipline.

Software Engineering (1990s)

Perhaps the most visible professional activity in IT is the construction of software systems. The term “software engineering” was coined in 1968 to name a discipline that would develop reliable and dependable complex software through rigorous engineering practice. Over the years, software engineers have developed practices in abstractions, specifications, languages, information-hiding, object-orientation, reuse, system scaling, exception handling, project management, measurement, testing, debugging, configurations, and documentation, and they have developed powerful tools and methods to support the practices (Meyer 2001).

Some observers do not think software engineers have gone far enough. Terry Winograd (1997) worries that they do not pay enough attention to the human side of design, and that an important new field, software architecture, may emerge to meet the need. Michael Dertouzos (2001) suggests that the “users” will eventually revolt and force a human-centered design paradigm.

Software engineers have criticized the standard computing curriculum for its view of programs as mathematical functions and programming as a mathematical activity. They believe that this view fosters to software construction that does not work well for the large systems encountered in practice. Some of them believe that the practices of traditional computer science are not fully compatible with those of software engineering and have proposed to split software engineering off into separate departments and degree programs. Noting other dualities such as chemical engineering and chemistry, they ask, why not software engineering and computer science? (Parnas 1997, Denning 1998)

This difference of worldviews was came to public attention in 1999 when ACM and the IEEE Computer Society took different paths with respect to licensing of software engineers. IEEE, which identifies more closely with engineering disciplines, decided to support efforts to certify professional software engineers and to cooperate with state authorities in specifying requirements and exams for licensing. ACM, which identifies more closely with mathematics and science, decided not to cooperate with state licensing efforts; their leaders believe that the software engineering body of knowledge is immature and that a certification would give the public a false impression of the person’s ability to produce safe and dependable software systems.

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 and with the rigorous definition of the functions of digital circuits. A few were concerned with models to define precisely the design principles and to forecast system behavior. Everyone agreed that programming was primarily a job of implementing mathematical functions on computers.

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 IT Profession and will share a common scientific core (Denning 1989).

IT Profession (2000s)

The birth of an IT profession has become our most challenging chasm. Let us turn now to Practices, Applications, Innovation, and Boundaries, which are all central to professional thinking, and which will be needed to cross this chasm.

Practices

Practices are habits, routines, processes, and skills performed by individuals and groups from experience and with little or no thought (Spinoza 1997). Practices are “embodied” or “ready to hand” knowledge -- they enable us to get things done quickly, without reflection. 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. 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. This difference is why a body practices is an explicit criterion for a profession.

Professional competence is judged by observing a person's practices to determine what actions the person is capable of taking (Dreyfus 1992). Ethical behavior is a practice of conforming one's actions to preset community standards of right and wrong, integrity, and honesty. Innovations are shifts of practices that enable practitioners to be more productive in some way; until an idea is practiced, it is no innovation.

Practices are not just personal. They exist in communities of people, where they manifest themselves not only as shared conventions, etiquette, 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. Many professional communities also set standards of performance and maintain institutions that certify competence at different levels. In some cases, such as engineering, education, accounting, law, or medicine, certification can be quite specific and rigorous. These 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. On the one side are faculty who hear "competence" as a code word for vocational "training" and 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 and obsolescence, an affront to the university's mission of general education. They value "reflective action" more than "reflexive action." On the other side are faculty who advocate more proficiency-based courses -- in which students don't pass until they can demonstrate that they can act effectively with the material. This debate is the first sign of an important change in our understandings of the four foundations -- data, information, knowledge, and practice.

Applications

In most professions, practice appears not as a form of knowledge, but as application of theory. In computing, we use the term "applications" to refer to software systems that apply our technological principles in specific domains of science, medicine, engineering, and commerce.

Scientific applications include statistical analyzers, equation solvers, chemical bond analyzers, ground soil diffusion analyzers, and fluid flow solvers. Medical applications are programs and systems such as patient record managers, EKG analyzers, 3-D imaging systems for MRI scans, real-time monitoring systems for intensive-care patients, and expert systems for diagnosis and prescriptions. Engineering applications include computer-aided design systems, building structure analysis systems, and flight simulators. Commercial applications include graph generators, word processors, spreadsheets, database systems, accounting and payroll systems, report generators, and programming environments.

Applications bring professionals cheek to jowl with pragmatists. In fact, the words "applications" and "users" annoy these pragmatists: it sounds like we think their world is subservient to ours. The way they see it, their world drives ours.

Innovation

Innovation is the adoption of new practices by people in a community, enabling them to produce more value for themselves. Inventions and good ideas are not innovations. The Patent Office bristles with inventions that were never commercialized

-- inventions that never produced innovations. Many novices use the terms invention and innovation interchangeably, a practice that misleads them and prevents them from forming business plans capable of crossing the chasms they will face. Bob Metcalfe, the inventor of ethernet, understands this distinction well. In a 1999 interview, his young interlocutor exclaimed, “Wow, it was the invention of the ethernet that enabled you to buy your house in Boston’s Back Bay!” Metcalfe shot back: “No, the reason I was able to afford that house is that I sold ethernets for ten years!”

About sixty years ago, our forebears articulated a “pipeline model” for innovation. Vannevar Bush gave it voice in his famous essay, Science, the endless frontier. This model became the basis of public policy for federal sponsorship of research in universities. According to this model, innovations result ultimately from ideas created by researchers or inventors; these ideas flow through a pipeline comprising stages of peer review, prototype development, manufacturing, and marketing, with only the best ones reaching consumers as products. This model places a great deal of value on free and open exchange of ideas. Although science in action has never followed this model (Latour 1987), it remains popular among academics and policy wonks.

During the 1990s, however, another model emerged -- actually, if you take a longer historical view, it reemerged -- the “marketplace model.” According to this model, entrepreneurial groups experiment with technology, seeking to develop prototypes for market as quickly as possible. They place a great deal of value on transforming people’s practices through new products and services. Many intend to sell their technology to a larger company rather than develop their own customer base.

The flow times in the idea-pipeline model are a few decades and, in the marketplace model, a few years. In 2000, the US federal government funded most of the idea-pipeline research through federal grants in universities totaling approximately $10B, while venture capitalists funded entrepreneurs at the much higher level of approximately $50B. It is no wonder that the rate of innovation in IT seemed to explode in the 1990s, that most of the innovations came from non-academic sources, and that most academicians were surprised at their own lack of advance knowledge of many innovations. Both models are a reality of innovation and must be understood by professionals and taught in universities.

Dennis Tsichritzis, the Chairman of GMD, the German National Research Center for Information Technology, argues that innovation is the ultimate objective of research (1997). He identified four major processes of innovation, each supported by its own kind of research:

Generating new ideas. Powerful new ideas shift the discourse, in turn shifting the actions of those practicing the discourse. Research consists of formulating, validating, and disseminating 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.

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 and in designing exercises to teach the practices. It places a great deal of emphasis on understanding that produces competence.

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.

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 roles and identities in the world. It places a great deal of emphasis on business model, market identity, position, and exploring marginal practices.

Tsichritzis explicitly advocates the first three processes as the portfolio of a 21st century research center (1997). Slywotzky advocates the fourth (1995). The first process is characteristic of the pipeline model, the last two of the marketplace model, and the second process bridges the two models. Traditional computer science places the most value on the first of these four processes. The IT profession will treat them equally.

Boundaries

We have noted a tendency for the IT specialties to be insular, trying to be self contained and minimizing interactions among themselves and with other fields. They are loath to cross the boundaries between them. This is a paradox given that those who crossed the boundaries have produced so much of the innovation we see around us.

The newspapers and technical magazines are filled with stories about new technologies that arise in collaborations between people from different fields. Look at the steady stream of IT inventions in medicine, libraries, business, e-commerce, biotech, entertainment, transportation, astronomy, telecommunications, science, and banking. Look at the success of interdisciplinary research groups like the Santa Fe Institute, government research labs, and supercomputing centers. Look at the many interdisciplinary programs promoted by the federal research agencies. IT professionals need to become proficient and comfortable with interdisciplinary teams exploring boundaries of their fields.

Computer science itself originated at the boundaries between electronics, science, and the mathematics of logic and calculation. Although many areas from algorithms to operating systems have developed into thriving scientific specialties, it would be a mistake to think we have run out of new boundaries that have the potential to change the field. Today's boundaries include

- New computing technologies including DNA, analog silicon, nanodevices, organic devices, and quantum devices;
- Internet computations mobilizing hundreds of thousands of computers. Recent examples are SETI.com, which explores for extra-terrestrial signals, and Juno.com, which mobilizes thousands of subscriber workstations into massively parallel supercomputers.
- 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. (Digital Libraries and Napster are examples.)
- Workflow and coordination technologies from the business workplace, where improving productivity is a constant concern.

These boundaries are the most likely sources of radical innovations. They are likely to yield new standard practices and core principles for information technology 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 than this. It is linked to enterpreneurship and the dynamics of professions (Spinoza 1997). Recall that one overarching responsibility of any profession is to take care of recurring breakdowns. Breakdowns attract entrepreneurs, who often find the seeds of solutions in anomalous practices that do not resonate with the current common sense of the field. The practices may be central in another field. Many entrepreneurs achieve their success by appropriating practices across the boundaries with other fields and transforming them to the center of the entrepreneur's field.

A short story will illustrate these statements. Early in the 1980s researchers in high-energy physics established bulletin board services to exchange preprints of their research 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 (file transfer protocol) connection to the server containing the paper, download a copy, close the connection, and read the file with a local word processor -- not exactly convenient. In the late 1980s, Tim Berners-Lee, then 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, who were already used to writing technical papers in TeX (Knuth's markup language). Berners-Lee and his colleagues called their network of hyperlinked documents the World Wide Web (1996a, 1996b).

In the early 1990s, Marc Andreesen 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 (Hafner 1996). 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 the company that eventually became Netscape. The browser revolutionized the Internet, transforming it into a household word and making it respectable to place "http://" addresses on every business card and advertisement. Andreesen was an entrepreneur who transformed an anomalous practice into a central one, thereby resolving the breakdown that motivated him.

It is no accident that Andreesen'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. At Intel, Andy Grove follows similar practices to foster innovation (1996).

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 early 2001 there were an estimated 300 million persons using the Web from 150 million computers offering well over 3 billion 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.

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.

Disappearing Dichotomies

We expend much energy in our field debating apparent distinctions in an effort to find better ways to approach research, product development, and education. The framework for an IT profession, sketched above, resolves five dichotomies that computing technologists struggle with today.

Computer Science v. X, for X being any of the IT-specific or IT-intensive areas. Within the framework of an IT profession, these areas are brothers and sisters in the same family. They have the same scientific core, but different practices. It is not necessary to figure out which one is better than the others.

Research v. Application. From the perspective of an IT profession, research seeks to produce innovations that resolve recurrent breakdowns, anticipate future breakdowns, and bring value to communities of people. Research in this sense is a blend of "basic" and "applied" and can be conducted in either an idea-pipeline and a market model of innovation. Much innovation flows from the boundaries, where the practices of information technologists interact with the practices of other domains. What is today called "application" is part of a process of building tools to facilitate transfer of practices across domain boundaries. Applications are essential to the innovations sought by researchers.

Researcher v. Practitioner. Professional societies in IT (e.g., ACM, IEEE, AAAI, SIAM) tend to categorize people as "researchers", "practitioners", or "users". These designations reflect the idea-pipeline model of innovation (researchers at the start, practitioners in the middle, users at the end). They rankle many pragmatists, who see their concerns and willingness to spend money, not researchers' ideas, as the driving forces of the technology. Researchers, inventors, practitioners, users, pragmatists, and users are all full partners in part of an IT profession.

Education v. Training. Learning the professional practices of an IT specialty at increasing levels of competence is every bit as important as learning the scientific and technological principles. The mark of a well-educated professional will be a balance of the two.

General v. Professional Education. General education seeks to produce a graduate who can act effectively by reading, writing, speaking, and listening, and who understands history, literature, art, sciences, philosophy, language, and social relationships. General education is the context in which a person can attain higher levels of professional competence.

The IT Schools Movement

On the campuses, there is a new movement to organize professional IT schools. The movement is gaining a momentum that overcomes the traditional territorialism of academic departments. The movement is propelled by three new realities: (1) IT is a profession of many specialties. Education of IT professionals can no longer be the responsibility of a single university department or degree program. (2) IT curricula must include a professional body of knowledge complementing the intellectual body of knowledge. Such an expansion is recognized by new guidelines of CSAB (Computer Science Accrediting Board) and by some national professional groups such as the British Computing Society. (3) Many universities have declared that they will be leaders in educating IT workers.

Until 2000, colleges of computing and information technology were few in number. The pioneers include the School of Information Technology and Engineering at George Mason University (1985), the School of Computer Science at Carnegie-Mellon University (1988), the College of Computing at Georgia Institute of Technology (1991), the College of Information Science and Technology at University of Nebraska Omaha (1996), and the College of IT at the United Arab Emirates University (2000). In 2000, the Computing Research Association and the Association for Computing Machinery sponsored the formation of a community of IT deans, which numbered about three dozen when the new programs were included. Also in 2000, the first model curriculum for an IT college appeared (Denning 2000).

Several of these schools are using a novel academic structure that is likely to appeal to a great many universities. An exemplar is the School of Informatics at Indiana University, which does not operate as a completely self-contained unit. They have a small core faculty and they partner with the faculty of participating departments from other schools. Their common core program in information technology includes segments on computer science, information science, public policy, business, and applications. Each participating department offers a specialization track for students who have completed the core. With this structure, students can achieve solid grounding in information technology, which they can then combine with business, law, health, or humanities.

In another decade, we can expect to see many universities offering degrees with models such as these. It is only a matter of time until the professional societies offer guidelines on curricula for IT schools. These schools will become the educational backbone for new entrants into the IT profession.

These schools may offer a way to address the perplexing problem of poor cooperation between university and industry education. There is a vast network of over 1600 corporate universities whose annual spending equals that in the public universities -- and yet there almost no interaction between them and the public universities in regard to curriculum, professional degrees, and continuing education. (Meister 1998) Universities and businesses have difficulties agreeing on joint research projects because of intellectual property issues. Many business leaders are loath to support public academic programs, believing that the state is already paying for that from their taxes.

When the Crossing is Complete

Most of those who use computers and communications do so through hardware, software, and networks whose inner workings are mysteries to them. These people seek professional help in taking care of their concerns about safe, reliable operations of these technologies. They expect professionals to be responsive, competent, ethical, and able to anticipate future breakdowns. Although an IT profession is an historical necessity, it has been slow to form because of limitations in our own conception of technology, customers, and profession. Many business leaders, for example, see improved standards of performance as an impediment to hiring IT workers.

The education of IT professionals will undergo at least four major changes. First, IT professionals will adopt a lifetime learning model for the profession, a model that includes IT learning before, during, and after college. Second, the important role of the vast network of corporate universities will be recognized. Third, IT schools will blossom on campuses. They will feature a common core curriculum serving a diverse mixture of IT-intensive specialties, and innovative models of cooperation among diverse departments with IT interests. Fourth, the conception of knowledge itself will change in IT schools, putting professional embodied practice on an equal footing with intellectual knowledge. The apprentice-master model will become more popular for teaching. IT professionals to design, build, test, and validate systems.

Research will change as the field shifts from the limited view that innovation depends on generating new ideas to a wider view that also encompasses marketplace processes of innovation. Universities will recognize both the idea-pipeline and the marketplace models of innovation. There will be a resurgence of interest in research at the boundaries between IT technical disciplines and other disciplines.

Professional societies worldwide will cooperate on the development of IT as a profession by sponsoring conferences, criteria for levels of professional competence, joint programs benefiting members of many societies at once, and models for various aspects of the lifelong learning process of IT professionals. With their help IT will develop a coherent identity that includes IT core fields, IT intensive fields, and IT infrastructure service fields, and will attract men and women from a much wider range of backgrounds. IT professionals from many specialties will proudly proclaim their allegiance to the profession as a whole as well as their own disciplines.

IT professionals will become much better listeners for the concerns of their clients. They will appreciate how IT adds value to their customers' lives. IT professionals will become much more human-centered in their approaches to design. The notion of developing technology because it is intellectually interesting will take second place to developing technology that creates value for people.

The path to this future will not be easy. Interests resisting change will throw up obstacles. Even so, there are already encouraging signs of progress.

The previous struggles of computing to cross smaller chasms broadened the discipline and prepared it for its most challenging chasm: the new profession. We are ready and we are starting to move. Who said crossing a chasm is easy?

References

Berners-Lee, Tim. 1996a. "The Web Maestro: An interview with Tim Berners-Lee." *Technology Review* (July).

Berners-Lee, Tim. 1996b. "WWW: Past, Present, and Future." *IEEE Computer 29*, 10 (Oct), 69-77.

Denning, Peter, David Brandin, and Daniel McCracken. 1979. "An ACM Executive Committee position on the crisis in experimental computer science." *ACM Communications* (Sept).

Denning, Peter. 1980. "What is experimental computer science?" *ACM Communications* (Oct).

Denning, Peter. 1981a. "Performance Analysis: Experimental Computer Science at its best." *ACM Communications* (Nov).

Denning, Peter., *et al.* 1981b. "A discipline in crisis -- the Snowbird Report." *ACM Communications* (June).

Denning, Peter, *et al.* 1989. "Computing as a discipline." *ACM Communications* (January 1989). *IEEE Computer* (February).

Denning, Peter. 1991. "Computing, Applications, and Computational Science," *ACM Communications 34*, 10 (Oct), 129-131.

Denning, Peter. 1997. "How we will learn." In *Beyond Calculation: The Next 50 Years of Computing* (P. Denning and R. Metcalfe, eds), Copernicus Books.

Denning, Peter. 1998. "Computer Science and Software Engineering: Filing for Divorce?" *ACM Communications 41* (July).

Denning, Peter, et al. 2000. *A Model Curriculum for an IT College.* Available as a web report <cne.gmu.edu/pjd/UAE>.

Dertouzos, Michael. 2001. *The Unfinished Revolution*. Harper Collins.

Dreyfus, Hubert. 1992. *What Computers Still Can't Do.* MIT Press.

Feldman, Jerome, *et al.* 1979. "Rejuvenating experimental computer science -- A report to the National Science Foundation and others." *ACM Communications* (Sept).

Grove, Andy S. 1996. *Only the Paranoid Survive.* Currency Doubleday.

Hafner, Katie., and Mathew Lyons. 1996. *Where Wizards Stay Up Late: The Origins of the Internet*. Simon and Schuster.

Harel, David. 2000. *Computers Ltd.: What They Really Can't Do.* Oxford University Press.

Hartmanis, Juris, *et al.* 1992. *Computing the future*. National Academy Press. (http://www.nas.edu)

Holmes, Neville. 2000. "Fashioning a Foundation for the Computing Profession". IEEE *Computer* (July). 97-98.

Indiana University. 2001. School of Informatics Website <informatics.indiana.edu>.

Latour, Bruno. 1987. *Science in Action*. Harvard University Press.

Meister, Jeanne. 1998. *Corporate Universities.* McGraw-Hill.

Meyer, Bertrand. 2001. "Software Engineering in the Academy." IEEE *Computer* (May), 28-35.

Moore, Geoffrey. 1991. *Crossing the Chasm*. Harvard Business.

National Research Council. 1999. *Being Fluent in Information Technology* (Larry Snyder, Panel Chair). National Academy Press.

Parnas, David. 1997. "Software Engineering: An unconsummated marriage." Inside RISKS column, *ACM Communications 40*, (September), 128.

Perelman, Lewis. 1992. *School's Out.* Avon.

Schneiderman, Ben. 1998. "Related-Create-Donate: An educational philosophy for the cyber-generation. *Computers & Education 31*, 1, 25-39.

Slywotzky, Adrian. 1995. *Value Migration.* Harvard Business School Press.

Snyder, Larry, *et al.* 1994. *Academic careers for experimental computer Scientists*. National Academy Press. (http://www.nas.edu)

Spinoza, Charles, Fernando Flores, and Hubert Dreyfus. 1997. *Disclosing New Worlds*, MIT Press.

Tachibana, Takashi. 1998. "Closing the knowledge gap between the scientist and nonscientist," Science 281 (7 Aug), 778-779.

Tichy, Walter. 1998. "Should Computer Scientists Experiment More?" *IEEE Computer* (May).

Tsichritzis, Dennis. 1997. "The dynamics of innovation." In *Beyond Calculation: The Next 50 Years of Computing* (P. Denning and R. Metcalfe, eds). Copernicus Books.

Winograd, Terry. 1997 "Interaction Design." In *Beyond Calculation: The Next 50 Years of Computing* (P. Denning and R. Metcalfe, eds). Copernicus Books.

Wulf, William. 1998. "The Urgency of Engineering Education Reform." *The Bridge 28*, 1 (Spring).

Zelkowitz, Marvin, and Dolores Wallace. 1998. "Experimental models for validating technology." *IEEE Computer* (May).