The Great Principles of Computing

Peter Denning teaches students at the Naval Postgraduate School how to develop strategic, big-picture thinking about the field of computing.

Peter Denning, a past president of ACM (1980-82), has been involved with communicating our discipline, computing, to outsiders since 1970. Along the way he invented the working set model for memory management, developed the theory of virtual memory, promulgated operating systems theory, co-invented operational analysis of system performance, co-founded CSNET, and led the ACM Publications Board while it developed the Digital Library. He is an ACM Fellow and holds five major ACM awards. He just completed a five-year term as chair of the ACM Education Board.

UBIQUITY: Talk a little about the Naval Postgraduate School, and your role there.

DENNING: Its primary job is to offer master's degrees and some PhDs to military officers from all the services. The Navy runs it as their "corporate university," but officers of the Air Force, Army, Marines, and allied armed services in 53 countries also attend, as well as a few DOD civilians. They come here for an intensive computer science graduate program that lasts four 12-week quarters a year. It's a demanding, full-time, yearlong masters program.

UBIQUITY: Are they good students?

DENNING: They're outstanding. Most are five to ten years out into their careers. The Navy offers graduate education as an incentive for valued officers to stay in the service. Then they try and make use of their education in their future military careers. They are disciplined, hard-working, forthright, smart, and intensely pragmatic.

UBIQUITY: Do they all have similar undergraduate preparation? Or is there some variety?

DENNING: There's a huge variety among our computer science students. Some of them have formal computing backgrounds; some are IT practitioners without a formal degree in an IT field; some come from electrical and other engineering disciplines; quite a few were educated in other fields at the undergraduate level, but now seek a master's degree in computer science. Many of these people have not been near a college campus for ten years and sometimes more, so they take advantage of a service we offer called the Refresher Quarter -- a ninth quarter preceding the ones devoted to the degree. Popular refresher topics are math, calculus, and programming. The Refresher Quarter also helps them re-acclimate to the academic environment.

UBIQUITY: How many students are there?

DENNING: This quarter there are about 1,500 in the entire Naval Postgraduate School. That's going to jump to about 1,800 within the next two quarters, when 300 Air Force students come. The computer science department by itself has about 120 students pursuing a master's and PhD degrees in computer science, another 120 pursuing in software engineering, and another 20 in Modeling of Virtual Environments and Simulations (MOVES). About 250 faculty teach these students. The low student-faculty ratio guarantees plenty of individual attention, especially when students are working on their masters theses.

UBIQUITY: How do you do research with such a relatively small PhD population?

DENNING: Every master’s student is required to do a master’s thesis, a year-long project. It gives the students the chance to propose, analyze, and study something that may provide an innovation for future defense operations. An “army” of 750 students completing master’s theses every year is a formidable force for innovation. Many faculty plot research agendas in the form of questions that can be investigated in one year; with such planning their research programs accomplish as much or more than smaller cadres of PhDs in civilian universities.

UBIQUITY: What is your title?

DENNING: I have two titles. I came in December 2002 as Chairman of the Computer Science Department. A few months later I also became Director of Cebrowski Institute for information innovation and superiority. It’s named after retired Admiral Art Cebrowski, who’s now directing the Office of Force Transformation for DOD. He is a great strategist and the architect of “force transformation”, which is the idea that the military needs to transform the way it thinks and acts for warfare in the years ahead. He works directly for Secretary Rumsfeld. The Institute carries his name to symbolize the type of work we do; he is not directly involved. My job is to coordinate faculty and students from all over the school as they do research relating to that topic.

UBIQUITY: What institutions have you been affiliated with?

DENNING: My undergraduate degree is from Manhattan College and my graduate degrees from MIT. My first faculty position was at Princeton. I was at Purdue in the 1970s, NASA Ames in the 1980s, George Mason University in the 1990s, and then here at the Naval Postgraduate School.

UBIQUITY: How does NPS compare to the other places?

DENNING: What's unique here is the atmosphere created by the combination of the military command-and-control environment on the one side, and the traditional academic consensus-oriented environment on the other. The two sides are in what feels to me like a very creative and productive tension. If the academic side gets too slow in deliberating something important to the military leadership, the military leadership will just start making it happen. It's that simple. So the faculty never spends a long time deliberating or thinking all the nuances of a project. Faculty initiatives that support the NPS missions are encouraged and welcomed. Sometimes the faculty pushes back on proposals from the military leadership, and the leadership listens and adjusts.

UBIQUITY: That must be academically refreshing.

DENNING: Exactly. We have an action-oriented environment. That we have DOD customers for research and curricula is something we never forget. Few curricula get stale. When we want to get things done around here, we get them done in a hurry.

UBIQUITY: Example?

DENNING: A year ago, in the first month after my arrival here, I asked our faculty to undertake a major curriculum review. By the end of January 2003 the project was in full swing. Our revision was organized around the theme of Great Principles of Computing. The faculty did the entire process -- talking through alternatives, planning the new courses, updating and revising all the core courses, and working out the annual schedule -- in six months, from start to finish. Students started on the new curriculum on October 1st. The very notion of completing a major curriculum revision in six months is close to unthinkable in any university I've been in. But we did it here. And the faculty approved it unanimously to boot. That’s one of the things I like about this place: we can accomplish major things in a short time.

UBIQUITY: Is that typical of how things work at the NPS?

DENNING: Yes. Another example is that, last year, we started a master’s program in homeland security. This was a rapid response to a new need for education in the new Department of Homeland Security. Already several dozen students have come into that program. Not only did we get a new program started in short order, we turned to innovative practices for executing it. To reduce the impact on the schedules of the people taking the program, we set it up so that the participants come to NPS for one intensive week at the start of every quarter. The rest of the quarter, they interact in study groups and with the instructors by electronic mail and the Web.

UBIQUITY: When one thinks of military officers, one thinks of leadership. Does the special interest of the students in leadership give the academic programs a special character?

DENNING: Yes, it does. Our students are professional leaders. Many of my students are Navy Commanders and Lieutenant Commanders, or Army and Marine Corps Majors and Captains. These people have had considerable experience being leaders of groups in their services. Several of my students served as Executive Officers on ships before coming here. These people are no-nonsense, very practical. They want to achieve results. They want us to give them the simplest and most direct tools needed to get the job done. They’re highly disciplined. They know how to produce actions. They also demand a lot of relevance.

UBIQUITY: Talk about those demands for relevance.

DENNING: Relevance means clear and direct value to the service missions these people expect on graduation. My students at George Mason might tolerate my launching into an abstract philosophical discussion about computing. But here, I'm likely to be soon interrupted by someone wanting to know how this helps them. My own path into computing -- and my ongoing interests in computing -- included a lot of mathematics. This is appropriate; computing is a very mathematical discipline. But many of my NPS students are not as excited about math as I am. If I don't demonstrate the value and relevance of a mathematical concept, I'll hear a voice saying, "Sir, why are we talking about this? What does this have to do with anything?"

UBIQUITY: When would that kind of question typically arise?

DENNING: Here's one that came up last quarter. All computer scientists hear about Alan Turing and his machine model of computation postulated in 1936. Not long into my discussion of the Turing machine, a student asked, "Sir, why are we talking about Turing machines? Why should we care about this?" I answered: "This is very fundamental to our understanding of computation. Every computer scientist needs to know about it." The student pressed on: "But, sir, when I get into my next command, how will my knowledge of Turing machines help?" This is challenging!

UBIQUITY: How do you approach the teaching of the Great Principles of Computing?

DENNING: One of our new courses is the Great Principles of Computing, which we offer to first-quarter students. It has a noble purpose: to introduce the field in terms of its fundamental principles rather than its core technologies. It serves as a road map to the rest of the curriculum. It starts to develop strategic, big-picture thinking about our field, thinking needed by these officers after finishing here. The idea of getting directly at the principles of what's going on underneath computing is very appealing to this group, rather than getting bogged down, as they like to call it, in the weeds. The challenge for me as a teacher is recognize what they will see as weeds. For my 35 years as a teacher of computing, Turing machines have looked pretty fundamental; to many of these officers, Turing machines look like weeds. We are therefore finding other ways to explain the limits of computing systems without taking them through Turing machine theory. I know that many newcomers to computing are interested to understand the essence of the field; they are not interested in the mathematical or technological intricacies. They much prefer trying to get their heads around five categories of computing laws and four of computing practices, rather than thirty fast-changing core technologies.

UBIQUITY: Any other reasons for focusing on the Great Principles?

DENNING: Yes, the next, after reducing the apparent complexity of the field, is the perception of computing as a field of programmers. In 1988, I led an ACM/IEEECS task force that produced the report *Computing as a Discipline*. We were very concerned about dispelling the image of computing as a field of programmers. Computing people know that programming is a small fraction of all we do; there's a lot of scientific and engineering content in computing that is not programming. Unfortunately, our attempt to change this perception did not succeed. Today the perception that computing is basically a field of programmers is as strong as ever.

UBIQUITY: Why didn't it succeed?

DENNING: I think it failed for a very practical reason. Everyone, from employers to students, sees computing as so useful, they want from the beginning to do useful things with the concepts they learn. Programming is the most direct path to this end. So, in most academic departments, the first course in computer science aims to teach concepts of computing and demonstrate them with hands-on programming. We have historically taught programming side-by-side with the basic concepts of computing. In the 1970s there was no standard for the first language – Fortran, Algol, MAD, LISP, PL/I, and a few others were common. In the 1980s we so liked Pascal for its simplicity and cleanness, it became a de facto standard. But many industry people complained that few workplaces use Pascal and urged us to work with the “real” languages of the workplace. Many faculty switched over to C, then C++, and more recently Java as the primary language for the first and second courses. These industry-strength languages involve many professional practices that make them quite complex. They are sophisticated languages and are not easily mastered in one or two semesters.

UBIQUITY: How has the switch to C++ and Java brought new problems?

DENNING: Students get consumed with trying to master the mechanics of programs. They are so busy trying to make programs work, they get sidetracked from understanding the underlying concepts of their programs. Their perception is that “Computer Science 1” is really “Programming” and not concepts of computing. The complexity of these languages has become a monster that we don’t know how to cope with. One of the consequences is a high dropout rate from our first courses -- about 30 to 50 percent. Another is that plagiarism and copying of programs is a major industry as students “do anything” to pass those courses. Thus for most students it looks like programming is the make-or-break entrée to the field; there’s no way in unless you can program. Our first course is a traumatic experience for many students. It’s no surprise to me that so many people have the impression that programming and computing are synonymous.

This issue is spilling down to high schools. A few years ago, the Educational Testing Service came under the same pressures as academic departments. They decided to switch the Advanced

Placement Curriculum to object oriented programming with Java as the language. Even though ETS has been moving slowly toward the implementation, this change brought much consternation of gnashing of teeth to high-school teachers. They were not trained in object-oriented programming and found learning it to be as time consuming and traumatic as do students in the first university courses.

UBIQUITY: What could you do to change that?

DENNING: That’s a tough one. Discussing the first language is a religious experience at most faculty meetings. No one can agree on what to change. I think the key is to treat programming as a primary practice of computing rather than a conceptual framework of computing. When we think of it as a skill, we would approach it like we do other skills: we proceed in steps up a ladder of increasing competence until we attain the competence level we seek. Applying this idea, we would start with simple languages and programs and get to the more complex ones later once students have developed enough skill. One colleague drew an apt analogy with the Psychology Department. Although graduates may be expected to have clinical skills, they are not expected to have those skills at the end of the first course: Psychology 101 is not organized to teach any clinical practice. By analogy, why do we feel compelled to teach an industry-strength language when none of those who complete the first course will be qualified as programmers anyway? The bottom line for me is, let’s treat programming as a practice and lay out coursework that progresses from simple languages to the complex in stages that students can cope with. My opinion appears to be in a minority right now.

UBIQUITY: Well what is the basis of their disagreement?

DENNING: Here’s how it is put to me: Java and C++ are the main languages out there in the world. It takes a while to learn languages these days because they are complex. If you’re going to go to the trouble of learning a language, why not learn one that might be useful?

UBIQUITY: Where does this issue stand now at the Naval Postgraduate School?

DENNING: First and foremost, our faculty agreed to approach programming as a core practice of computing and not to intermingle it in the same courses with the concepts of computing. The job of the programming practices courses is to teach programming and help students become competent at it. In fact, we decided to recognize a distinction between practice and principles throughout our graduate curriculum. We put programming practices into a three-course sequence (under that name) and let students immerse themselves in the mechanics and other practical details of getting programs to work correctly. In addition to programming, we identified three other core practices of computing; we teach them all under the heading of “computing practices”.

UBIQUITY: What are the core practices?

DENNING: Programming, systems, modeling, and innovating.

UBIQUITY: Talk a little bit about each practice, starting with programming.

DENNING: Let me back up to the notion of core practice. At NPS, we are using practice in the sense of embodied knowledge – things we know how to do because they are trained into us, things we can do without conscious thought. They are habits, routines, repetitive procedures, automatic responses, and the like. They can only be known through our performances, which can be at any of the levels of beginners, advanced beginners, competent, proficient, expert, virtuoso, master, or legend. A core practice is one that is essential to be a computing professional; we say you can’t be a complete professional without minimal competence in these four. Back to your question about programming. Everybody agrees: if you’re going to be a computer person, you need to be able to program, and you must be able to do it in multiple languages. Associated with this is a mental practice we call “algorithmic thinking” – a way of approaching problems by designing algorithms to find the solution. Our clients and customers expect us to be facile with the many computer languages that exist. If we say we are not, they will respond: “How can you call yourself a computing professional?”

UBIQUITY: What do you mean by systems as a practice?

DENNING: Systems are coordinated collections of hardware and software components that work together to provide stated services. Systems practice refers the engineering process of putting together requirements, specifications, prototypes, and tests; of evaluating costs, benefits, risks, and safety; of anticipating what can go wrong; of creating the overall architecture that guides the integration of components; and of finding “emergent” behaviors that can not be predicted from examination of individual components. These practices are central to software engineering. Systems practice is not the same as programming, although it’s close to the flavor called “programming in the large”. Associated with this is a mental practice called systems thinking.

UBIQUITY: Talk about modeling as a practice.

DENNING: Information systems are often so complex that we can’t completely understand them without being able to perform experiments, collect data, and interpret the data. We often make architectural decisions based on preliminary experiments to see what works or what doesn’t work. We use data gathered from carefully designed experiments to validate system correctness and predict future performance. The modeling practice refers to the use of tools to design experiments, gather data, test hypotheses, express models, and predict performance. Associated with this is a mental practice for dealing with uncertainty by using random variables to describe systems.

UBIQUITY: Talk about the fourth practice, innovation.

DENNING: This is an interesting one. Computing people are expected to help other people promote changes in their organizations or in the way they do things through the use of computing. Change leaders are celebrated in annual issues of leading technology magazines. Computing professionals are also expected to take leadership in promoting those changes. We believe the ability to produce innovation is a core practice of computing. In the NPS context this is very appropriate because the military services want their officers to be change leaders. The Chief of Naval Operations wants all personnel to participate in a “culture of innovation.” All this belies the stereotype of the military as a change-averse organization. Associated with this is a mental practice of seeing opportunities for innovation in the problems that people experience, and seeing how to transform their social practices to overcome the problems.

UBIQUITY: Talk about the Principles of Computing that you helped develop?

DENNING: There are two categories of principles -- mechanics and design. Mechanics refers to the fundamental laws and recurrences of computation. Mechanics subdivides into five categories; we call them computation, communication, coordination, recollection, and automation. Design refers to process -- conventions we use to organize ourselves to build good computing systems -- and to architecture -- conventions for arranging the components of systems.

UBIQUITY: Start by describing the three “c”s -- computation, communication and coordination.

DENNING: Computation addresses the time and storage requirements to various kinds of problems, along with practical and theoretical limitations on what problems computing machines can solve. Communication addresses how to represent and transmit data, receive transmissions, and deal with noise in the channels. Coordination addresses issues of multiple entities cooperating to achieve a common result. Humans, who coordinate all the time, pass on their coordination problems to machines when they delegate computations to them. Coordination addresses computer-supported cooperative work (CSCW), human computer interaction (HCI), and concurrency control.

UBIQUITY: What do you mean by recollection?

DENNING: Recollection addresses the issues of storing and recalling information or data. It involves memory systems, database systems and storage systems. The World Wide Web, in its role as a global hyperlinked information system, is the most expansive example of recollection.

UBIQUITY: Finally, describe the principle of automation.

DENNING: Automation concerns the general question of what cognitive tasks, ordinarily associated with human intelligence, can be automated. In many ways, “What can be automated?” is the fundamental question behind all computing. The automation area of computing includes software agents, artificial intelligence, and cognitive science.

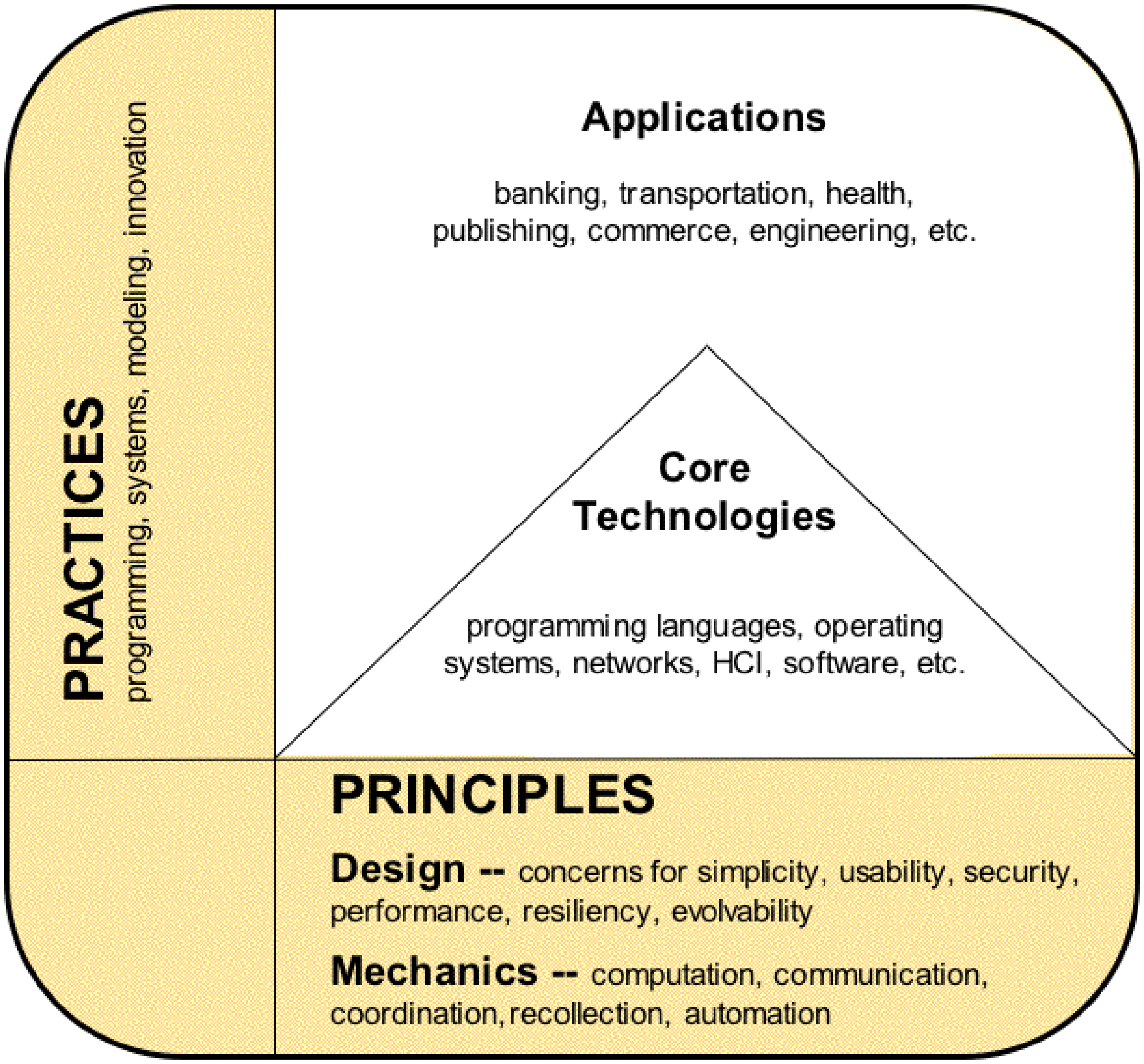
UBIQUITY: Is this complete? Are their any other categories of computing mechanics?

DENNING: Not as far as we can tell. We made a list of the core technologies of computing -- programming languages, operating systems, databases, graphics, software engineering, and so on down a list of 30 -- and we found elements of all five categories in every one. These five categories covered everything we saw in the core technologies.

UBIQUITY: What about design? How does that fit in?

DENNING: The principles of design are motivated by pervasive, major concerns for simplicity, performance, resilience, evolvability, and security. Design refers to two levels of arrangements, both aimed at building systems that are effective at meeting these concerns and deliver pleasing and proper results. One we call architecture, which is how we arrange components of a system; the other we call process, which is how we arrange ourselves, our workflows, and our thinking. Examples of architectural principles are modularity, abstraction, information hiding, and virtual machines. Examples of process principles are reusability, rapid prototyping, human-centered methodology, and process-centered methodology. These principles are not fundamental laws of nature; instead they are more like social conventions we have adopted over the years to help us produce good software. We bring design to our mechanical systems to produce effective computations.

This graphic shows how all these parts fit together. The Great Principles framework consists of principles and practices along separate dimensions, supporting core technologies of computing, which in turn support applications. The principles are of two kinds: mechanics -- how computations and computers work -- and design -- how to build them to work well. These principles pervade core technologies that in turn support many application areas. Practices, the embodied skills of computing professionals, apply at all levels.


PRACTICES
programming, systems, modeling, innovation
Applications
banking, transportation, health, publishing, commerce, engineering, etc.
Core Technologies
programming languages, operating systems, networks, HCI, software, etc.
PRINCIPLES
Design -- concerns for simplicity, usability, security, performance, resiliency, evolvability
Mechanics -- computation, communication, coordination, recollection, automation

UBIQUITY: How does this approach mesh with the ACM?

DENNING: The last time ACM expressed itself on a structural model of the field was in the 1989 report called "Computing as a Discipline." I was chair of the committee that did that. We proposed a matrix model with 9 rows and 3 columns. The rows named 9 core technologies such as algorithms and data structures, numerical computation, and operating systems. The columns named theory, abstraction, and design, the three processes that pervade thought and practice in computing. We defined a core technology as a self-sustaining area, supporting many applications, with its own theory, abstraction, and design, and its own literature, conferences, and professional groups. Three difficulties have arisen with that way of describing the field. The number of core technology areas has exploded from nine to over thirty; extending the model becomes very unwieldy. The second difficulty is that most people no longer make sharp distinctions among the processes of theory, abstraction, and design. That model doesn't fit the way many people see computing these days. The third difficulty is that computing joins telecommunications and organizational dynamics as the basis of the Information Technology field; the matrix model of computing does not extend to these other fields. The ACM more recently engaged in Computing Curriculum 2001, which was a joint effort with IEEE. This project focused on identifying a common core of computing that supported software engineering, information systems, and computer engineering as well as traditional computer science. The result was much less structured than the 1989 model, but much more representative of IT. The Great Principles framework gives a way to organize the core ideas of Curriculum 2001.

UBIQUITY: If you could do a controlled experiment, with equivalent students and equivalent teachers, and run them through a curriculum as you are proposing versus a typical curriculum existing today, would you see any difference in the student outputs?

DENNING: Yes, in a number of ways. One is that these students be able to distinguish between practice and principle, between what they do with their hands and what they do with their minds.

Second, they would be generally stronger than most other graduates today in the area of modeling, which in some curricula is hardly covered. Third, more people would appreciate and embrace programming’s multilingual nature. Fourth, our graduates would be able to participate in and contribute to a culture of innovation. Fifth, our graduates would be able to see the field with a simpler structure, which would help them interpret trends in the world and decide how to approach them.

UBIQUITY: Give a real-life example of how that would work.

DENNING: A good test of the great principles structure is how it would inform us about important emerging trends such as self-aware computing, mobile computing, and nanotechnology. Suppose someone asks me, “What do you think the role of nanotechnology will be in the future of computing? Would it be a new category of mechanics?” I might observe that the basic issue of nanotechnology is to drastically shrink the size of computing and mechanical devices, without adding new functions. The same principles of computation, communication, coordination, recollection, and innovation would apply -- no change in the model there. We might well expect significant changes in practice ranging from new fabrication principles, to new applications, to new designs. There might even be a new area of practice for nanotechnology. All this is consistent with the framework.

UBIQUITY: What is the next step? Do you think that you could develop a textbook that could be widely accepted? Do you think you could influence other programs?

DENNING: I am working on a textbook that would focus mainly on the conceptual material, the five categories of mechanics and the design principles, for people who want to understand that part and organize courses around it. But I don’t think a textbook is enough. We need to rethink our approach to teaching computing practices and certifying the levels of skill in them. We also need to encourage new storytellers to come forward with compelling stories about how technologies and principles developed in computing in its various categories. We need storytellers with the skills of Richard Feynman in physics, Carl Sagan in Astronomy, or Bob Hazen in general science. Might we interest James Burke in examining computing in these terms on popular TV? One can dream.

UBIQUITY: Yes. Dreams are good.

DENNING: An important step is to work with the ACM Education Board and its counterpart in the IEEE Computer Society, to use this framework to improve the structure and execution of the Curriculum 2001. It would be particularly useful to distinguish between computing practices and computing principles. There are no incompatibilities between the Curriculum 2001 recommendations and the great principles framework.

UBIQUITY: That sounds like quite an effort.

DENNING: Indeed. For innovations to stick, they need to be institutionalized. We need support structures and agreed-on standards. The ACM can help with that. It’s well worth the effort.

END

This is the first part of a two-part interview, the second part of which will be posted in March.

More on the Naval Postgraduate School can be found at http://www.nps.navy.mil/.