Computer Science: An Interview

Peter J. Denning, *Naval Postgraduate School*
Neville Holmes, *University of Tasmania*

Computing has its own paradigm, distinct from engineering or science.

This e-mail interview with Peter Denning sprang from comments in the October 2010 The Profession column, "The Future of the Computing Profession: Readers' E-mails" about Denning's essay, "The Great Principles of Computing" (*American Scientist*, Sept./Oct. 2010, pp. 369-372; tinyurl.com/2dtzcdv).

SCIENCE AND ENGINEERING

Question: Remind me of the purpose of your essay.

Response: I wrote for an audience of scientists from many fields. My purpose was to show them that the computing field has reached maturity and must be taken seriously in science and engineering. In fact, there is a strong case that computing is a great domain of science, alongside the traditional domains of the physical, life, and social sciences.

I appreciated your suggestion to your readers that they should read the article. Computing professionals need not be defensive about their field and can instead focus on bringing their computing expertise into the service of the science and engineering fields they work with.

Question: The illustration for your essay showed several science figures and one engineering figure floating in a cloud, with the computing figure rowing a cloud canoe to join them. Does that engineering figure really belong among the science figures?

Response: That figure is an editorial artist's interpretation of the message of the article. The artist was trying to reflect my message that computing is growing up and joining the ranks of mature science and engineering. That's why the artist probably thought it well to include an engineering figure. Perhaps a better figure would have been a cloud marked "computing," with all the other fields frantically rowing canoes to catch it.

Question: What is the relationship between science and engineering in computing?

Response: The science aspect of computing emphasizes the experimental method for making discoveries and validating hypotheses. The engineering aspect emphasizes design and implementation processes to manage costs and minimize failure risks. Both aspects rely on the same body of knowledge.

However, these two aspects do not fully characterize computing. There is a third aspect—the practice of interpreting the world as information processes and solving problems by finding computational steps and methods to explain and control those processes. Today, many people call this aspect "computational thinking," a term that to my mind is too narrow because it overlooks "computational doing," that is, the professional practice of computing. The computational approach can be found in diverse science areas, for example, DNA translation, where it has led to important discoveries; and in diverse engineering areas, for example, finite element structural design, where it has enabled the construction of very reliable, complex systems. Computing definitely brings something new to the party.

Question: In the academic world, computing professionals get started in either computer science or computer engineering departments. Is that the best way to organize the curriculum?

Response: It's more complicated than that. Students start their professional careers in other programs as well, including software engineering, information systems, information science, and information technology. Some schools have tried to blur the distinction between computer science and electrical and computer engineering by creating combined EECS departments. Some of these degree programs are in schools of science, some in schools of engineering,

Continued on page 94

 Published by the IEEE Computer Society 0018-9162/11/$26.00 © 2011 IEEE

Continued from page 96

and some in the new generation of schools of computing. I favor schools of computing.

Question: Where does software engineering fit?

Response: That continues to be a matter of debate. The founders of that field envisioned bringing rigorous engineering design processes to software so that software systems would be predictable and fault tolerant, and act within tolerances. There are many today who think this goal hasn't been accomplished. Despite its flaws, both schools of engineering and schools of science claim software engineering. In some schools, software engineering is part of computer science; in others, it's separate.

In computing, we have both a scientific and an engineering paradigm, and it's often hard to pry them apart.

Question: Algorithms, processes, and computations all seem like abstract, mathematical entities, whereas the practice of computing seems like design, implementation, and testing. Does that not make computing essentially an engineering field in its practice?

Response: No. As computing links up with numerous other fields, computing professionals will have to learn some of the domain knowledge of the other fields. Many of those fields require extensive domain knowledge of experimental methods, experiment design, and data analysis. Computing professionals who specialize in engineering design would have difficulty contributing to those fields. In fact, they often have difficulty contributing to engineering because the newer, evolutionary approaches to system development rely heavily on experimental methods to evaluate prototypes and decide which design alternative to explore next.

Question: Henry Petroski, a regular columnist for *American Scientist*, wrote a column for *IEEE Spectrum* titled "Engineering Is Not Science" (tinyurl.com/PtrskSp). He said, "Confusing the two keeps us from solving the problems of the world." Are computing professionals therefore scientists?

Response: No. As I stated, some computing professionals emphasize engineering practice, some science practice, and some both together. They all emphasize computational practice.

Petroski is making a political argument—offering advice for politicians who don't make clear distinctions about engineering and science. They need engineering but ask for science to solve big problems. Most fields of engineering say they are based on some form of science, thus they devote some time to understanding that science. I suppose an outsider who doesn't understand the subtleties might think science gestates engineering.

In computing, we have both a scientific and an engineering paradigm, and it's often hard to pry them apart. For example, we benchmark systems and networks by running workloads on them and measuring their throughput and response time. Then we take those results and design new systems and networks that can meet throughput and response time targets. We use the experimental method to parameterize the engineering design. I no longer think that it's a productive argument to try to separate these two aspects of computing. We have our own paradigm, and it mixes the traditional paradigms in new ways.

Question: If computing has its own paradigm, doesn't that imply that computing is neither science nor engineering, but rather a field distinct from either? This idea would seem to be supported by the exploitation of computing by many other disciplines, such as medicine and the performing arts.

Response: My sentiment exactly. I've written about this twice in my column (tinyurl.com/pjd09sep; tinyurl.com/pjd09dec).

SCIENCE AND MATHEMATICS

Question: Mathematics is often described as the handmaiden of the sciences. Does this mean that mathematics is distinct from the sciences?

Response: The relation of mathematics to engineering and the sciences is an ongoing philosophical debate. Mathematics is concerned with finding provably true statements (theorems) about the relationships among entities. Both engineering and science use mathematics for the models of the systems they're building or phenomena they're exploring. The models enable predictions, which scientists can validate with experiments and engineers can use to find tolerances for their systems. However, just because both use mathematics doesn't mean they are subsets of mathematics, any more than it means they contain mathematics.

Question: How is computer science distinct from mathematics?

Response: Computing has brought a new dimension to this. Whereas mathematics is a language for *describing* realities, computing is a means of *generating* them. One of the first questions explored in computing was how mathematical functions could be calculated. A program describes a method for evaluating a function, whereas the computation from executing that program generates the results. When a computing professional describes object-oriented design to a physicist, the latter is likely to point out that computing has no special claim to abstraction; physicists have been using abstractions for years. In response, the computing

professional is likely to point out that objects generate processes that conform to the abstractions. Computing abstractions do things.

Question: When I was an engineering student long, long ago, pure mathematics and applied mathematics were compulsory and separate subjects for the first two years of study. Pure mathematics was about the theory of mathematics, and applied mathematics was about the computational use of mathematics—a use that projected strongly into third- and fourth-year subjects in various branches of engineering. What distinguishes computing from applied mathematics?

Response: While it's true that some computations are designed by applying mathematics, others are designed from computational thinking without recourse to mathematics. For example, when I build a spreadsheet that calculates throughput and response time for a system's queuing model, I'm applying the mathematics of queues. On the other hand, when I design a workflow system that recognizes speech acts and tracks their commitments to completion, I'm using computational thinking. Much of computing doesn't apply any known mathematics or even try to develop new mathematics.

COMPUTING AND PEOPLE

Question: Computing today is about more than crunching numbers and manipulating symbols. As people increasingly need and want to use computers, that is, to interact with them personally, hasn't cognitive science become as relevant to computing professionals as mathematics?

Response: First, I agree that computing is so much more than numbers and symbols. Computing's biggest use today is in systems of communication and coordination. Many people today see their computers and the Internet as a critical communication system they can't do without. This has opened up new worlds of connections between computing professionals and other people. Just take a look at social networking and the new branch of research called network science. I agree that some of these connections will bring computing professionals in contact with cognitive science and that cognitive science will influence many directions in computing. However, I would not want to imply that every computing professional needs to learn cognitive science any more than network science.

Question: Doesn't all this imply that computer science should have a strong component of ethics and sociology?

Response: Here I'd like to answer by applying some of the distinctions you frequently champion. Let's distinguish a discipline (a field of study like computer science) and a profession (a set of skilled practitioners who serve clients in a domain). Professions have codes of ethics because it's important to society that their clients can trust them to exercise their special skills in beneficial ways. Professional societies such as the ACM and the IEEE Computer Society support professionals in many ways, including providing a code of ethics and curriculum recommendations for supporting disciplines such as computer science and engineering.

So, in answer to your question, yes, the computing profession needs a code of ethics. Computing curricula accreditation requirements include attention to ethics because the curricula are the initial education of many computing professionals. I recommend reading "A Mature Profession of Software Engineering," a marvelous paper by Gary Ford and Norman Gibbs written in 1996 but quite useful today (tinyurl.com/gfng96rp).

Question: You drew my attention to an interesting guest blog on the online version of *Scientific American* (tinyurl.com/WartikScAm). In it, Steve Wartik expresses the opinion that "we would be best served by viewing Computer Science as a branch of philosophy." His main point is that computer science researchers do more philosophizing than experimenting. Is this a valid argument?

Response: That was his punch line. His argument was that computer scientists don't do enough experimental work to validate many of their claims, so it looks to him like a lot of theory without much grounding in practice. That's when he says it looks more like philosophy than science.

While I can see why Wartik might say that, I don't accept his argument. It doesn't agree with the evidence I'm seeing. I see deep, fundamental principles in all parts of computing (greatprinciples.org). Just because Microsoft Windows doesn't seem principled to him, doesn't mean there are few operating systems principles. I'm seeing an explosion of interest in experimental methods to validate claims, not just among researchers but also among practitioners—consider Google's emphasis on data.

Question: And in conclusion?

Response: It looks to me that computing has its own paradigm distinct from engineering or science, that computing has deep principles that were not known in any field of engineering or science, and that as the field matures, our practitioners are increasingly involved with experimental methods as well as design.

I'm proud to be called a computer scientist and computing professional.

Peter J. Denning *is a distinguished professor of computer science at the Naval Postgraduate School, Monterey, California. He is also the director of the Cebrowski Institute for Innovation and Information Superiority and a past president of the ACM. Readers will find more of his writings online at cs.gmu.edu/cne/pjd/PUBS. Contact him at pjd@nps.edu.*

Neville Holmes *is an honorary research associate at the University of Tasmania's School of Computing and Information Systems. Contact him at neville.holmes@utas.edu.au.*