DOI:10.1145/1516046.1516054

Peter J. Denning

The Profession of IT
Beyond Computational Thinking

If we are not careful, our fascination with "computational thinking" may lead us back into the trap we are trying to escape.

In the midst of our struggle to better articulate why computing is so much broader than programming, a movement of sorts has emerged. It is being called "computational thinking."[8] The U.S. National Science Foundation's Computer and Information Science and Engineering (CISE) directorate has asked most proposers, especially those in its CPATH initiative, to include a discussion of how their projects advance computational thinking. Carnegie Mellon University's Center for Computational Thinking says, "It is nearly impossible to do research in any scientific or engineering discipline without an ability to think computationally....[We] advocate for the widespread use of computational thinking to improve people's lives."[1]

Computational thinking is seen by its adherents as a novel way to say what the core of the field is about, a lever to reverse the decline of enrollments, and a rationale for accepting computer science as a legitimate field of science. This movement is driven by four main concerns:

- Bringing computer science to the table of science (as partner, not programmer).
- Finding ways to make computer science a more attractive field for students to major in and for other sciences to collaborate with.
- Resurrecting ongoing inquiry into the deep questions of the field.[6,9]
- Showing that computation is fundamental, and often unavoidable, in most endeavors—a desire to proselytize.

Since starting a stint at NASA-Ames in 1983, I have been heavily involved with computational science and I have devoted a substantial part of my own career to advancing these objectives. Since

2003 I have advocated a great-principles approach to the perennially open question, "What is computer science?"[4]

Yet I am uneasy. I am concerned that the computational thinking movement reinforces a narrow view of the field and will not sell well with the other sciences or with the people we want to attract. I worry that we are not getting out of the box, but are merely repackaging it with new paper and a fresh ribbon.

In this column, I will examine two key questions:

- Is computational thinking a unique and distinctive characterization of computer science?
- Is computational thinking an adequate characterization of computer science?

My own conclusion is that both answers are no. I will suggest that a principles-based framework answers both questions yes. We are custodians of a deep and powerful discourse: Let's not hide it with an inadequate name.

What is Computational Thinking?

Computational thinking has a long history within computer science. Known in the 1950s and 1960s as "algorithmic thinking," it means a mental orientation to formulating problems as conversions of some input to an output and looking for algorithms to perform the conversions.

Today the term has been expanded to include thinking with many levels of abstractions, use of mathematics to develop algorithms, and examining how well a solution scales across different sizes of problems.[1]

Is Computational Thinking Unique to Computer Science?

In the 1940s, John von Neumann wrote prolifically on how computers would be not just a tool for helping science, but a way of doing science.

As early as 1975, Physics Nobel Laureate Ken Wilson promoted the idea that simulation and computation

were a way to do science that was not previously available. Wilson's Nobel Prize was based on breakthroughs he achieved in creating computational models whose simulations produced radical new understandings of phase changes in materials. In the early 1980s, Wilson joined with other leading scientists in many fields to advocate that the grand challenges of science could be cracked by computation and that the government could accelerate the process by supporting a network of supercomputing centers.[7] They argued that computation had become a third leg of science, joining the traditional legs of theory and experiment. The term "computational thinking" was common in their discussions.

The computational sciences movement eventually grew into a huge interagency initiative in high-performance computing, and culminated in the U.S. Congress passing a law funding a high-performance computing initiative in 1991.

This movement validated the notion that computation (and computational thinking) is essential to the advancement of science. It generated a powerful political movement that codified this notion into a U.S. federal law.

It is important to notice that this movement originated with the leaders of the physical and life sciences. Computer science was present but was not a key player. Computer scientists, in fact, resisted participation until NSF CISE and DARPA set up research programs open only to those collaborating with other sciences.

Computation is unavoidable not only in the method of study, but in what is studied.

In the middle 1980s, Ken Wilson advocated the formation of departments of computational science in universities. He carefully distinguished them from computer science. The term "computational science" was chosen to avoid confusion with computer science.

Thus, computational science is seen in the other sciences not as a notion that flows out of computer science, but as a notion that flows from science itself. Computational thinking is seen as a characteristic of this way of science. It is not seen as a distinctive feature of computer science.

Therefore, it is unwise to pin our hopes on computational thinking as a way of telling people about the unique character of computer science. We need some other way to do that.

The sentiment that computational thinking is a recent insight into the true nature of computer science ignores the venerable history of computational thinking in computer science and in all the sciences. Computer science is a science in its own right (see the sidebar "Computer Science as Science").

Is Computational Thinking Adequate for Computer Science?

In 1936 Alan Turing defined what it means to compute a number. He offered a model of a computing machine and showed that the machines were universal (one could simulate another). He then used his theory to settle a century-old "decision problem" of mathematics, whether there is a by-inspection method to tell if a set of decision rules can terminate with a decision in a finite number of moves. He showed that the "decision problem" was not computable and argued that the very act of inspecting is inherently computational: not even inspectors can avoid computation. Computation is universal and unavoidable. His paper truly was the birth of computer science.

The modern formulations of science

Computer Science as Science

Since its beginnings in the late 1930s, computer science has been a unique combination of math, engineering, and science. It is not one, but all three. Major subsets form legitimate fields of math, engineering, or science. But if you focus on a single subset, you cannot express the uniqueness of the field.

The term "computer science" traces back to the writings of John von Neumann, who believed that the architecture of machines and applications could be put on a rigorous scientific basis.

Until about 1990, the emphasis within the field was developing and advancing the technology. Building reliable computers within a networking infrastructure was a grand challenge that took many years. Now that this has been accomplished, we are increasingly able to emphasize the experimental method and reinvigorate our image as a science. Our many partnerships with other sciences including biology, physics, astronomy, materials science, economics, cognitive science, and sociology, have led to amazing innovations.

These collaborations have uncovered questions in the other fields about whether computer science is legitimately science. Many see computer people as engineers implementing principles they did not discover rather than equal partners in the search for new principles. So it matters whether computer science qualifies as a full-fledged science. Whether a field is seen as a science depends on its satisfying six criteria:[5]

- **Has an organized body of knowledge**
- **Results are reproducible**
- **Has well developed experimental methods**
- **Enables predictions, including surprises**
- **Offers hypotheses open to falsification**
- **Deals with natural objects**

Computer science easily passes the first five of these tests, so the debate has tended to center on the last. During the past decade, prominent scientists in other fields have discovered natural information processes—affirming the sixth criterion.[3] The older definition of computer science as "the study of phenomena surrounding computers," which dates back to Alan Perlis, George Forsythe, and Allen Newell around 1970, is giving way to "the study of information processes, natural and artificial." The shift from computer as object of study to computer as tool is enabling us to revisit the deep questions of our field in the new light of computation as a lens through which to see the world. The most fundamental of these questions is: What is computation?[6,9]

recognize the same truth when they say that computation is an essential method of doing science. In fact, a growing number of scientists are now saying that information processes occur naturally (for example, DNA transcription) and that computation is needed to understand and eventually control them.[3] So computation is unavoidable not only in the method of study, but in what is studied.

This is a subtle but important distinction. Computation is present in nature even when scientists are not observing it or thinking about it. Computation is more fundamental than computational thinking. For this reason alone, computational thinking seems like an inadequate characterization of computer science.

A number of us developed a great principles framework that exposes the fundamental scientific principles of computing[4,6] (see the sidebar "The Great Principles Framework"). This framework interprets computer science as the study of fundamental properties of information processes, both natural and artificial. Computers are the tool, not the object of study. Computation pervades everyday life.[2]

The great principles framework reveals that there is something even more fundamental than an algorithm: the representation. Representations convey information. A computation is an evolving representation and an algorithm is a representation of a method to control the evolution.

In this framework, computational thinking is not a principle; it is a practice. A practice is a way of doing things at which we can develop various levels of skill. Computational thinking is one of several key practices at which every computer scientist should be competent (see the sidebar "The Great Principles Framework"). It shortchanges computer science to try to characterize the field by mentioning only one essential practice without mentioning the others or the principles of the field.

The real value of computer science is in the offers we are able to make from our expertise, which is founded in a rich and deep discourse.

Conclusion

Computation is widely accepted as a lens for looking at the world. We do not need to sell that idea. Computational thinking is one of the key practices of computer science. But it is not unique to computing and is not adequate to portray the whole of the field.

In the 1960s and 1970s we allowed, and even encouraged, the perception "CS = programming," which is now to our dismay widely accepted outside the field and is connected with our inability to take care of the concerns listed at the beginning of this column. But given the outside perception, computational thinking is all too easily seen as a repackaging—a change of appearance but not of substance. Do we really want to replace that older notion with "CS = computational thinking"? A colleague from another field recently said to me: "You computer scientists are hungry! First you wanted us to take your courses on literacy and fluency. Now you want us to think like you!"

I suggest that the real value of computer science is in the offers we are able to make from our expertise, which is founded in a rich and deep discourse. We are valued at the table when we help the others solve problems they care about. We are most valued not for our computational thinking, but for our computational doing. C

References

1. Carnegie Mellon University Center for Computational Thinking; http://www.cs.cmu.edu/~CompThink.
2. Computer Science Unplugged Web site; http://csunplugged.org.
3. Denning, P. Computing is a natural science. *Commun. ACM 50*, 7 (July 2007), 13–18.
4. Denning, P. Great principles of computing. *Commun. ACM 46*, 11 (Nov. 2003), 15–20.
5. Denning, P. Is computer science science? *Commun. ACM 48*, 4 (Apr. 2005), 27–31.
6. Great Principles of Computing Web site; http://greatprinciples.org.
7. Wilson, K.G. Grand challenges to computational science. In *Future Generation Computer Systems*. Elsevier, 1989, 171–189.
8. Wing, J. Computational thinking. *Commun. ACM 49*, 3 (Mar. 2006), 33–35.
9. Wing, J. Five deep questions in computing. *Commun. ACM 51*, 1 (Jan. 2008), 58–60.

Peter J. Denning (pjd@nps.edu) is the director of the Cebrowski Institute for Information Innovation and Superiority at the Naval Postgraduate School in Monterey, CA, and is a past president of ACM.

The Great Principles Framework

The Great Principles (GP) framework is a way to express computer science as a field of science based on deep and enduring fundamental principles.[3,4,6] The framework has two parts: core principles and core practices.

The core principles are statements and stories about the immutable laws and recurrences that shape and constrain all computing technologies. They can be grouped into seven categories:

- **Computation**
- **Communication**
- **Coordination**
- **Recollection**
- **Automation**
- **Evaluation**
- **Design**

These are not mutually exclusive groups of principles, but windows that bring particular perspectives about computing. The Internet, for example, is a technology that draws its operating principles primarily from communication, coordination, and recollection, and its architecture from design and evaluation.

The core practices are areas of skill and ability at which computing people can display various levels of performance such as beginner, competent, and expert. There are four core practices:

- **Programming**
- **Engineering of systems**
- **Modeling**
- **Applying**

Computational thinking can be seen either as a style of thought that runs through the practices or as a fifth practice. It is the ability to interpret the world as algorithmically controlled conversions of inputs to outputs.