The Profession of IT, Beyond Computational Thinking

Denning, Peter J.

Beyond Computational Thinking (June 2009) If we are not careful, our fascination with "computational thinking" will lead us back into the trap we are trying to escape.

http://hdl.handle.net/10945/35494
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.” 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.”

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

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?”

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:
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. 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.

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.

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

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
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.

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. The framework has two parts: core principles and core practices.

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.

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.