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

*Downloaded from NPS Archive: Calhoun*
viewpoints

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.”

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.
- 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?”

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.

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
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 study, but in each the emphasis is on the nature of computer science ignores the unique combination of computer science. We need some other way to do that. The notion that computation 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”).

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
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. 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 computing4,6 (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.

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.

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.

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.

References


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

Copyright held by author.