Computational thinking, 10 years later

March 24 2016, by Jeannette M. Wing

"Not in my lifetime." That's what I said when I was asked whether we would ever see computer science taught in K-12. It was 2009, and I was addressing a gathering of attendees to a workshop on computational thinking convened by the National Academies.

I'm happy to say that I was wrong.

It's been 10 years since I published my three-page "Computational Thinking" viewpoint in the March 2006 issue of the *Communications of the ACM*. To celebrate its anniversary, let's consider how far we've come.

Think back to 2005. Since the dot-com bust, there had been a steep and steady decline in undergraduate enrollments in computer science, with no end in sight. The computer science community was wringing its hands, worried about the survival of their departments on campuses. Unlike many of my colleagues, I saw a different, much rosier future for computer science. I saw that computing was going to be everywhere.

I argued that the use of computational concepts, methods and tools would transform the very conduct of every discipline, profession and sector. Someone with the ability to use computation effectively would have an edge over someone without. So, I saw a great opportunity for the computer science community to teach future generations how computer scientists think. Hence "computational thinking."

I must admit, I am surprised and gratified by how much progress we

have made in achieving this vision: Computational thinking will be a fundamental skill used by everyone in the world by the middle of the 21st century. By fundamental, I mean as fundamental as reading, writing and arithmetic.

The third pillar of the scientific method

I knew that in the science and engineering disciplines, computation would be the third pillar of the scientific method, along with theory and experimentation. After all, computers were already used for simulation of large, complex physical and natural systems. Sooner or later, scientists and engineers of all kinds would come to recognize the power of computational abstractions, such as algorithms, data types and state machines.

And today, with the advent of massive amounts of data, researchers in all disciplines—including the arts, humanities and social sciences—are discovering new knowledge using computational methods and tools.

In the past 10 years, I visited nearly 100 colleges and universities worldwide and witnessed a transformation at the undergraduate level. Computer science courses are now offered to students who are not majoring in computer science. These courses are not computer programming courses, but rather focus on core concepts in computer science. At Harvard, this course (CS50) is one of the most popular courses not just on its campus but also at rival Yale's campus. And what about computer science enrollments? They are skyrocketing!

Perhaps the most surprising and gratifying result is to see what is happening at the K-12 level. First, the United Kingdom's grassroots effort Computing At School led the Department of Education to require computing in K-12 schools in England starting September 2014. The statutory guidance for the national curriculum says, "A high-quality

computing education equips pupils to use computational thinking and creativity to understand and change the world."

In addition, the BBC, in partnership with Microsoft and other companies, funded the design and distribution of the BBC micro:bit. One million of these small programmable devices will be distributed free this month: One for every 11-12 year-old (Year 7) student in the UK, along with their teachers. Microsoft Research contributed to the design and testing of the device; and the MSR Labs Touch Develop team provided a programming language and platform for the BBC micro:bit, as well as teaching materials.

Second, code.org is a nonprofit organization, started in 2013, dedicated to the mission of providing access to computer science education to all. Microsoft, along with hundreds of other corporate and organizational partners, helps sponsor the activities of code.org.

Third, internationally there is a groundswell interest in teaching computer science at the K-12 level. I know of efforts in Australia, Israel, Singapore and South Korea. China is likely to make a push soon, too.

Computer science for all

Most gratifying to me is President Barack Obama's pledge to provide $4 billion in funding for computer science education in U.S. schools as part of the Computer Science for All Initiative he announced on January 30. That initiative includes $120 million from the National Science Foundation, which will be used to train as many as 9,000 more high school teachers to teach computer science and integrate computational thinking into their curriculum.This push for all students to learn computer science comes partly from market demand for workers skilled in computing—from all sectors, not just information technology. We see this at Microsoft too; our enterprise customers in all sectors, such as

automotive, manufacturing and pharmaceutical, are coming to Microsoft because they need more computing expertise.

Still, practical challenges and research opportunities remain. The main practical challenge is that we do not have enough K-12 teachers trained to teach computer science to K-12 students. I am optimistic that, over time, we will solve this problem.

There also are interesting research questions that I would encourage computer scientists to pursue, working with the cognitive and learning sciences communities. First, what computer science concepts should be taught when, and how?

Consider an analogy to mathematics. We teach numbers to 5-year-olds, algebra to 12-year-olds and calculus to 18-year-olds. We have somehow figured out the progression of concepts to teach in mathematics, where learning one new concept builds on understanding the previous concept, and where the progression reflects the progression of mathematical sophistication of a child as he or she matures.

What is that progression in computer science? For example, when is it best to teach recursion? Children learn to solve the Towers of Hanoi puzzle (for small n) and in history class we teach "divide and conquer" as a strategy for winning battles. But is the general concept better taught in high school? We teach long division to 9-year-olds in 4th grade, but we never utter the word "algorithm." And yet the way it is taught, long division is just an algorithm. Is teaching the general concept of an algorithm too soon for a 4th grader? More deeply, are there concepts in computing that are innate and do not need to be formally learned?

Second, we need to understand how best to use computing technology in the classroom. Throwing computers in the classroom is not the most effective way to teach computer science concepts. How can we use

technology to enhance the learning and reinforce the understanding of computer science concepts? How can we use technology to measure progress, learning outcomes and retention over time? How can we use technology to personalize the learning for individual learners, as each of us learn at a different pace and have different cognitive abilities?

We have made tremendous progress in injecting computational thinking into research and education of all fields in the last ten years. We still have a ways to go, but fortunately, academia, industry and government forces are aligned toward realizing the vision of making computational thinking commonplace.

Provided by Microsoft

Citation: Computational thinking, 10 years later (2016, March 24) retrieved 28 June 2024 from https://phys.org/news/2016-03-years.html