JANUARY 10, 2014

COMPUTATIONAL THINKING BENEFITS SOCIETY

Jeannette M. Wing, Corporate Vice President, Microsoft Research

Fonte: http://socialissues.cs.toronto.edu/2014/01/computational-thinking/

Computer science has produced, at an astonishing and breathtaking pace, amazing technology that has transformed our lives with profound economic and societal impact. Computer science's effect on society was foreseen forty years ago by Gotlieb and Borodin in their book *Social Issues in Computing*. Moreover, in the past few years, we have come to realize that computer science offers not just useful software and hardware artifacts, but also an intellectual framework for thinking, what I call "computational thinking" [Wing06].

Everyone can benefit from thinking computationally. My grand vision is that computational thinking will be a fundamental skill—just like reading, writing, and arithmetic—used by everyone by the middle of the 21st Century.

This article describes how pervasive computational thinking has become in research and education. Researchers and professionals in an increasing number of fields beyond computer science have been reaping benefits from computational thinking. Educators in colleges and universities have begun to change undergraduate curricula to promote computational thinking to all students, not just computer science majors. Before elaborating on this progress toward my vision, let's begin with describing what is meant by computational thinking.

1. What is computational thinking?

1.1 Definition

I use the term "computational thinking" as shorthand for "thinking like a computer scientist." To be more descriptive, however, I now define computational thinking (with input from Al Aho at Columbia University, Jan Cuny at the National Science Foundation, and Larry Snyder at the University of Washington) as follows:

Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out.

Informally, computational thinking describes the mental activity in formulating a problem to admit a computational solution. The solution can be carried out by a human or machine. This latter point is important. First, humans compute. Second, people can learn computational thinking without a machine. Also, computational thinking is not just about problem solving, but also about problem formulation.

In this definition I deliberately use technical terms. By "expressing" I mean creating a linguistic representation for the purpose of communicating a solution to others, people or machines. The expressiveness of a language, e.g., programming language, can often make the difference between an elegant or inelegant solution, e.g., between a program provably absent of certain classes of bugs or not. By "effective," in the context of the Turing machine model of computation, I mean "computable" (or "decidable" or "recursive"); however, it is open research to revisit models of computation, and thus the meaning of "effective," when we consider what is computable by say biological or quantum computers [Wing08] or what is solvable by humans [Levin13, Wing08].

1.2. Abstraction is Key

Computer science is the automation of abstractions [1]. So, the most important and high-level thought process in computational thinking is the abstraction process. Abstraction is used in defining patterns, generalizing from specific instances, and parameterization. It is used to let one object stand for many. It is used to capture essential properties common to a set of objects while hiding irrelevant distinctions among them. For example, an algorithm is an abstraction of a process that takes inputs, executes a sequence of steps, and produces outputs to satisfy a desired goal. An abstract data type defines an abstract set of values and operations for manipulating those values, hiding the actual representation of the values from the user of the abstract data type. Designing efficient algorithms inherently involves designing abstract data types.

Abstraction gives us the power to scale and deal with complexity. Applying abstraction recursively allows us to build larger and larger systems, with the base case (at least for traditional computer science) being bits (0's and 1's). In computing, we routinely build systems in terms of layers of abstraction, allowing us to focus on one layer at a time and on the formal relations (e.g., "uses," "refines" or "implements," "simulates") between adjacent layers. When we write a program in a high-level language, we are building on lower layers of abstractions. We do not worry about the details of the underlying hardware, the operating system, the file system, or the network; furthermore, we rely on the compiler to correctly implement the semantics of the language. The narrow-waist architecture of the Internet demonstrates the effectiveness and robustness of appropriately designed abstractions: the simple TCP/IP layer at the middle has enabled a multitude of unforeseen applications to proliferate at layers above, and a multitude of unforeseen platforms, communications media, and devices to proliferate at layers below.

---

[1] Aho and Ullman in their 1992 *Foundations of Computer Science* textbook define Computer Science to be "The Mechanization of Abstraction."

2. Computational Thinking and Other Disciplines

Computational thinking has already influenced the research agenda of all science and engineering disciplines. Starting decades ago with the use of

computational modeling and simulation, through today's use of data mining and machine learning to analyze massive amounts of data, computation is recognized as the third pillar of science, along with theory and experimentation [PITAC 2005].

Consider just biology. The expedited sequencing of the human genome through the "shotgun algorithm" awakened the interest of the biology community in computational concepts (e.g., algorithms and data structures) and computational approaches (e.g., massive parallelism for high throughput), not just computational artifacts (e.g., computers and networks). In 2005, the Computer Science and Telecommunications Board of the National Research Council (NRC) published a 468-page report laying out a research agenda to explore the interface between biology and computing [NRC05]. In 2009, the NRC Life Sciences Board's study on Biology in the 21st Century recommends that "within the national New Biology Initiative, priority be given to the development of the information technologies and sciences that will be critical to the success of the New Biology [NRC09]." Now at many colleges students can choose to major in computational biology.

The volume and rate at which scientists and engineers are now collecting and producing data—through instruments, experiments, simulations, and crowd-sourcing—are demanding advances in data analytics, data storage and retrieval, as well as data visualization. The complexity of the multi-dimensional systems that scientists and engineers want to model and analyze requires new computational abstractions. These are just two reasons that every scientific directorate and office at the National Science Foundation participated in the Cyber-enabled Discovery and Innovation, or CDI, program, an initiative started when I first joined NSF in 2007. By the time I left, the fiscal year 2011 budget request for CDI was $100 million. CDI was in a nutshell "computational thinking for science and engineering [CDI11]."

Computational thinking has also begun to influence disciplines and professions beyond science and engineering. For example, areas of active study include algorithmic medicine, computational economics, computational finance, computational law, computational social science, digital archaeology, digital arts, digital humanities, and digital journalism. Data analytics is used in training Army recruits, detecting email spam and credit card fraud, recommending movies and books, ranking the quality of services, and personalizing coupons at supermarket checkouts. Machine learning is used by every major IT company for understanding human behavior and thus to tailor a customer's experience to his or her own preferences. Every industry and profession talks about Big Data and Cloud Computing. New York City and Seattle are vying to be named Data Science Capital of the US [Miller13].

3. Computational Thinking and Education

In the early-2000s, computer science had a moment of panic. Undergraduate enrollments were dropping. Computer science departments stopped hiring new

faculty. One reason I wrote my 2006 CACM article on computational thinking was to inject some positive thinking into our community. Rather than bemoan the declining interest in computer science, I wanted us to shout to the world about the joy of computing, and more importantly, about the importance of computing. Sure enough, today enrollments are skyrocketing (again). Demand for graduates with computing skills far exceeds the supply; six-figure starting salaries offered to graduates with a B.S. in Computer Science are not uncommon.

3.1 Undergraduate Education

Campuses throughout the United States and abroad are revisiting their undergraduate curriculum in computer science. They are changing their first course in computer science to cover fundamental principles and concepts, not just programming. For example, Carnegie Mellon revised its undergraduate first-year courses to promote computational thinking for non-majors [BryantSutnerStehlik10]. Harvey Mudd redesigned its introductory course with stellar success, including increasing the participation of women in computing [Klawe13]. At Harvard, "In just a few short years CS50 has rocketed from being a middling course to one of the biggest on campus, with nearly 700 students and an astounding 102-member staff [Farrell13]." For MIT's introductory course to computer science, Eric Grimson and John Guttag say in their opening remarks "I want to begin talking about the concepts and tools of computational thinking, which is what we're primarily going to focus on here. We're going to try and help you learn how to think like a computer scientist [GrimsonGuttag08]."

Many such introductory courses are now offered to or required by non-majors to take. Depending on the school, the requirement might be a general requirement (CMU) or a distribution requirement, e.g., to satisfy a science and technology (MIT), empirical and mathematical reasoning (Harvard), or a quantitative reasoning (Princeton) requirement.

3.2 What about K-12?

Not till computational thinking is taught routinely at K-12 levels of education will my vision be truly realized. Surprisingly, as a community, we have made faster progress at spreading computational thinking to K-12 than I had expected. We have professional organizations, industry, non-profits, and government policymakers to thank.

The College Board, with support from NSF, is designing a new Advanced Placement (AP) course that covers the fundamental concepts of computing and computational thinking (see the CS Principles Project). Phase 2 of the CS Principles project is in play and will lead to an operational exam in 2016-2017. Roughly forty high schools and ten colleges are part of piloting this course in the next three years. Not coincidentally, the changes to the Computer Science AP course are consistent with the changes in introductory computer science courses taking place now on college campuses.

Another boost is expected to come from the NSF's Computing Education for the 21st

Century (CE21) program, started in September 2010 and designed to help K-12 students, as well as first- and second-year college students, and their teachers develop computational thinking competencies. CE21 builds on the successes of the two prior NSF programs, CISE Pathways to Revitalized Undergraduate Computing Education (CPATH) and Broadening Participating in Computing (BPC). CE21 has a special emphasis on activities that support the CS 10K Project, an initiative launched by NSF through BPC. CS 10K aims to catalyze a revision of high school curriculum, with the new AP course as a centerpiece, and to prepare 10,000 teachers to teach the new courses in 10,000 high schools by 2015.

Industry is also promoting the importance of computing for all. Since 2006, with help from Google and later Microsoft, Carnegie Mellon has held summer workshops for high school teachers called "CS4HS." These workshops are designed to deliver the message that there is more to computer science than computer programming. CS4HS spread in 2007 to UCLA and the University of Washington. By 2013, under the auspices of Google, CS4HS had spread to 63 schools in the United States, 20 in China, 12 in Australia, 3 in New Zealand, and 28 in Europe, the Middle East and Africa. Also at Carnegie Mellon, Microsoft Research funds the Center for Computational Thinking, which supports both research and educational outreach projects.

Computing in the Core is a "non-partisan advocacy coalition of associations, corporations, scientific societies, and other non-profits that strive to elevate computer science education to a core academic subject in K-12 education, giving young people the college- and career-readiness knowledge and skills necessary in a technology-focused society." Serving on Computing in the Core's executive committee are: Association For Computing Machinery, Computer Science Teachers Association, Google,IEEE Computer Society, Microsoft, and National Center for Women and Information Technology.

Code.org is a newly formed public non-profit, sister organization of Computing in the Core. Its current corporate donors are Allen and Company, Amazon, Google, JPMorgan Chase and co., Juniper Networks, LinkedIn, Microsoft, and Salesforce. These companies and another 20 partners came together out of need for more professionals trained with computer science skills. Code.org hosts a rich suite of educational materials and tools that run on many platforms, including smart phones and tablets. It lists local high schools and camps throughout the US where students can learn computing.

Computer science has also gotten attention from elected officials. In May 2009, computer science thought leaders held an event on Capitol Hill to call on policymakers to make sure that computer science is included in all federally-funded educational programs that focus on science, technology, engineering and mathematics (STEM) fields. The U.S. House of Representatives designated the first week of December as Computer Science Education Week, originally conceived by Computing in the Core, and produced in 2013 by Code.org. In June 2013, U.S. Representative Susan Brooks (R-IN) and Representative Jared Polis (D-

CO) and others introduced legislation to bolster K-12 computer science education efforts. A month later, U.S. Senators Robert Casey (D-PA) and Marco Rubio (R-FL) followed suit with similar legislation.

Computational thinking has also spread internationally. In January 2012, the British Royal Society published a report that says that "'Computational thinking' offers insightful ways to view how information operates in many natural and engineered systems" and recommends that "Every child should have the opportunity to learn Computing at school." ("School" in the UK is the same as K-12 in the US.) Since that report the UK Department for Education published in February 2013 a proposed national curriculum of study for computing [UKEd13] with the final version of the curriculum becoming statutory in September 2014. In other words, by Fall 2014, all K-12 students in the UK will be taught concepts in computer science appropriate for their grade level. Much of the legwork behind this achievement was accomplished by the grassroots effort called "Computing at School." This organization is helping to organize the teacher training in the UK needed to achieve the 2014 goal.

Asian countries are also making rapid strides in the same direction. I am aware of efforts similar to those in the US and the UK taking place in China, Korea, and Singapore.

4. Progress So Far and Work Still to Do

Nearly eight years after the publication of my CACM Viewpoint, how far have we come? We have come a long way, along all dimensions: computational thinking has influenced the thinking in many other disciplines and many professional sectors; computational thinking, through revamped introductory computer science courses, has changed undergraduate curricula. We are making inroads in K-12 education worldwide.

While we have made incredible progress, our journey has just begun. We will see more and more disciplines make scholarly advances through the use of computing. We will see more and more professions transformed by their reliance on computing for conducting business. We will see more and more colleges and universities requiring an introductory computer science course to graduate. We will see more and more countries adding computer science to K-12 curricula.

We need to continue to build up and on our momentum. We still need to explain better to non-computer scientists what we mean by computational thinking and the benefits of being able to think computationally. We need to continue to promote with passion and commitment the importance of teaching computer science to K-12 students. Minimally, we should strive to ensure that every high school student around the world has access to learning computer science. The true impact of what we are doing now will not be seen for decades.

Computational thinking is not just or all about computer science. The educational benefits of being able to think computationally—starting with the use of abstractions—enhance and reinforce intellectual skills, and thus can be

transferred to any domain. Science, society, and our economy will benefit from the discoveries and innovations produced by a workforce trained to think computationally.

Personal Notes and Acknowledgements

Parts of this article, which I wrote for Carnegie Mellon School of Computer Science's publication *The Link* [Wing11], were based on earlier unpublished writings authored with Jan Cuny and Larry Snyder. I thank them for letting me them use our shared prose and for their own efforts in advocating computational thinking.

Looking back over how much progress has been made in spreading computational thinking, I am grateful for the opportunity I had while I was the Assistant Director of the Computer and Information Science and Engineering (CISE) Directorate of the National Science Foundation. I had a hand in CDI and CE21 from their start, allowing me—through the reach of NSF—to spread computational thinking directly to the science and engineering research (CDI) and education (CE21) communities in the US. Jan Cuny's initiative and persistence led to NSF's efforts with the College Board and beyond.

Since the publication of my CACM article, which has been translated into French and Chinese, I have received hundreds of email messages from people of all walks of life—from a retired grandfather in Florida to a mother in central Pennsylvania to a female high school student in Pittsburgh, from a Brazilian news reporter to the head of a think tank in Sri Lanka to an Egyptian student blogger, from artists to software developers to astrophysicists—thanking me for inspiring them and eager to support my cause. I am grateful to everyone's support.

Bibliography and Further Reading

Besides the citations I gave in text, I recommend the following references: CSUnplugged [BellWittenFellows10] for teaching young children about computer science without using a machine; the textbook used in MIT's 6.00 Introductory to Computer Science and Programming [Guttag13]; a soon-to-be published book on the breadth of computer science, inspired by Feynman lectures for physics [HeyPapay14]; a framing for principles of computing [Denning10]; and two National Research Council workshop reports [NRC10, NRC11], as early attempts to scope out the meaning and benefits of computational thinking.

[BellWittenFellows10] Tim Bell, Ian H. Witten, and Mike Fellows, "Computer Science Unplugged," http://csunplugged.org/, March 2010.

[BryantSutnerStehlik10] Randal E. Bryant, Klaus Sutner and Mark Stehlik, "Introductory Computer Science Education: A Deans' Perspective," Technical Report, CMU-CS10-140, August 2010.

[CDI11] Cyber-enabled Discovery and Innovation, National Science Foundation, http://www.nsf.gov/crssprgm/cdi/ , 2011.

[Denning10] Peter J. Denning, "The Great Principles of Computing," *American Scientist,* pp. 369-372, 2010.

[GrimsonGuttag08] Eric Grimson and John Guttag, *6.00 Introduction to Computer Science and Programming, Fall 2008*. (Massachusetts Institute of Technology: MIT OpenCourseWare). http://ocw.mit.edu (accessed January 3, 2014). License: Creative Commons Attribution-Noncommercial-Share Alike.

[Guttag13] John V. Guttag, *Introduction to Computation and Programming Using Python*, MIT Press, 2013.

[HeyPapay14] Tony Hey and Gyuri Papay, *The Computing Universe*, Cambridge University Press, scheduled for June 2014.

[Klawe13] Maria Klawe, "Increasing the Participation of Women in Computing Careers," Social Issues in Computing,http://socialissues.cs.toronto.edu/2013/12/women/, 2013.

[Farrell13] Michael B. Farrell, "Computer science fills seats, needs at Harvard," Boston Globe,http://www.bostonglobe.com/business/2013/11/26/computer-science-course-breaks-stereotypes-and-fills-halls-harvard/7XAXko7O392DiO1nAhp7dL/story.html, November 26, 2013.

[Levin13] Leonid Levin, "Universal Heuristics: How Do Humans Solve 'Unsolvable' Problems?," Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence Lecture Notes in Computer Science Volume 7070, 2013, pp 53-54.

[Miller13] Claire Cain Miller, "Geek Appeal: New York vs. Seattle," New York Timeshttp://www.nytimes.com/2013/04/14/education/edlife/new-york-and-seattle-compete-for-data-science-crown.html?_r=0, April 14, 2013.

[NRC05] Frontiers at the Interface of Computing and Biology, National Research Council, 2005.

[NRC09] "A New Biology for the 21st Century," National Research Council, 2009.

[NRC10] "Report of a Workshop on the Scope and Nature of Computational Thinking," National Research Council, 2010.

[NRC11] "The Report of a Workshop on Pedagogical Aspects of Computational Thinking, National Research Council, 2011.

[PITAC05] President's Information Technology Advisory Council, "Computational Science: Ensuring America's Competitiveness," Report to the President, June 2005.

[UKEd13] UK Department for Education, "Computing Programmes of study for Key Stages 1-4," February 2013,http://media.education.gov.uk/assets/files/pdf/c/computing%2004-02-13_001.pdf

[Wing06] Jeannette M. Wing, "Computational Thinking," Communications of the ACM, Vol. 49, No. 3, March 2006, pp. 33–35. In French: http://www.cs.cmu.edu/afs/cs/usr/wing/www/ct-french.pdf and in Chinese:http://www.cs.cmu.edu/afs/cs/usr/wing/www/ct-chinese.pdf

[Wing08] Jeannette M. Wing, "Five Deep Questions in Computing, Communications of the ACM, Vol. 51, No. 1, January 2008, pp. 58-60.

[Wing11] Jeannette M. Wing, "Computational Thinking: What and Why," The Link, March 2011.

Jeannette M. Wing is Corporate Vice President, Microsoft Research. She is on leave as President's Professor of Computer Science from Carnegie Mellon University where she twice served as Head of the Computer Science Department. From 2007-2010 she was the Assistant Director of the Computer and Information Science and Engineering Directorate at the National Science Foundation. She received her S.B.,S.M., and Ph.D. degrees from the Massachusetts Institute of Technology. Wing's general research interests are in formal methods, programming languages and systems, and trustworthy computing. She is a Fellow of the American Academy of Arts and Sciences, American Association for the Advancement of Science (AAAS), Association for Computing Machinery (ACM), and Institute of Electrical and Electronics Engineers (IEEE).