thelink.
The magazine of the Carnegie Mellon University School of Computer Science

Research Notebook: Computational Thinking--What and Why?

BY Jason Togyer - Sun, 2011-03-06 16:03

By Jeannette M. Wing

In a March 2006 article for the Communications of the ACM, I used the term "computational thinking" to articulate a vision that everyone, not just those who major in computer science, can benefit from thinking like a computer scientist [Wing06]. So, what is computational thinking? Here's a definition that Jan Cuny of the National Science Foundation, Larry Snyder of the University of Washington, and I use; it was inspired by an email exchange I had with Al Aho of Columbia University:

> *Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent. [CunySnyderWing10]*

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, or more generally, by combinations of humans and machines.

My interpretation of the words "problem" and "solution" is broad. I mean not just mathematically well-defined problems whose solutions are completely analyzable, e.g., a proof, an algorithm, or

a program, but also real-world problems whose solutions might be in the form of large, complex software systems. Thus, computational thinking overlaps with logical thinking and systems thinking. It includes algorithmic thinking and parallel thinking, which in turn engage other kinds of thought processes, such as compositional reasoning, pattern matching, procedural thinking, and recursive thinking. Computational thinking is used in the design and analysis of problems and their solutions, broadly interpreted.

The Value of Abstraction

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 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're building on lower layers of abstractions. We don't 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.

Computational thinking draws on both mathematical thinking and engineering thinking. Unlike mathematics, however, our computing systems are constrained by the physics of the underlying information-processing agent and its operating environment. And so, we must worry about boundary conditions, failures, malicious agents, and the unpredictability of the real world. And unlike other engineering disciplines, in computing --thanks to software, our unique "secret weapon"--we can build virtual worlds that are unconstrained by physical realities. And so, in cyberspace our creativity is limited only by our imagination.

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].

The expedited sequencing of the human genome through the "shotgun algorithm" awakened the interest of the biology community in computational methods, not just computational artifacts (such as computers and networks). The volume and rate at which scientists and engineers are now collecting and producing data--through instruments, experiments and simulations--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 participates in the Cyber-enabled Discovery and Innovation, or CDI, program, an initiative started four years ago with a fiscal year 2011 budget request of $100 million. CDI is in a nutshell "computational thinking for science and engineering."

Computational thinking has also begun to influence disciplines and professions beyond science and engineering. For example, areas of active study include algorithmic medicine, computational archaeology, computational economics, computational finance, computation and journalism, computational law, computational social science, and digital humanities. Data analytics is used in training Army recruits, detecting email spam and credit card fraud, recommending and ranking the quality of services, and even personalizing coupons at supermarket checkouts.

At Carnegie Mellon, computational thinking is everywhere. We have degree programs, minors, or tracks in "computational X" where X is applied mathematics, biology, chemistry, design, economics, finance, linguistics, mechanics, neuroscience, physics and statistical learning. We even have a course in computational photography. We have programs in computer music, and in computation, organizations and society. The structure of our School of Computer Science hints at some of the ways that computational thinking can be brought to bear on other disciplines. The Robotics Institute brings together computer science, electrical engineering, and mechanical engineering; the Language Technologies Institute, computer science and linguistics; the Human-Computer Interaction Institute, computer science, design, and psychology; the Machine Learning Department, computer science and statistics; the Institute for Software Research, computer science, public policy, and social science. The newest kid on the block, the Lane Center for Computational Biology, brings together computer science and biology. The Entertainment Technology Center is a joint effort of SCS and the School of Drama. SCS additionally offers joint programs in algorithms, combinatorics and optimization (computer science, mathematics, and business); computer science and fine arts; logic and computation (computer science and philosophy); and pure and applied logic (computer science, mathematics, and philosophy).

Computational Thinking in Daily Life

Can we apply computational thinking in daily life? Yes! These stories helpfully provided by Computer Science Department faculty demonstrate a few ways:

Pipelining: SCS Dean Randy Bryant was pondering how to make the diploma ceremony at commencement go faster. By careful placement of where individuals stood, he designed an

efficient pipeline so that upon the reading of each graduate's name and honors by Assistant Dean Mark Stehlik, each person could receive his or her diploma, then get a handshake or hug from Mark, and then get his or her picture taken. This pipeline allowed a steady stream of students to march across the stage (though a pipeline stall occurred whenever the graduate's cap would topple while getting hug from Mark).

Seth Goldstein, associate professor of computer science, once remarked to me that most buffet lines could benefit from computational thinking: "Why do they always put the dressing before the salad? The sauce before the main dish? The silverware at the start? They need some pipeline theory."

Hashing: After giving a talk at a department meeting about computational thinking, Professor Danny Sleator told me about a hashing function his children use to store away Lego blocks at home. According to Danny, they hash on several different categories: rectangular thick blocks, other thick (non-rectangular) blocks, thins (of any shape), wedgies, axles, rivets and spacers, "fits on axle," ball and socket and "miscellaneous." They even have rules to classify pieces that could fit into more than one category. "Even though this is pretty crude, it saves about a factor of 10 when looking for a piece," Danny says. Professor Avrim Blum overheard my conversation with Danny and chimed in "At our home, we use a different hash function."

Sorting: The following story is taken verbatim from an email sent by Roger Dannenberg, associate research professor of computer science and professional trumpeter. "I showed up to a big band gig, and the band leader passed out books with maybe 200 unordered charts and a set list with about 40 titles we were supposed to get out and place in order, ready to play. Everyone else started searching through the stack, pulling out charts one-at-a-time. I decided to sort the 200 charts alphabetically O(N log(N)) and then pull the charts O(M log(N)). I was still sorting when other band members were halfway through their charts, and I started to get some funny looks, but in the end, I finished first. That's computational thinking."

Benefits of Computational Thinking

Computational thinking enables you to bend computation to your needs. It is becoming the new literacy of the 21st century. Why should everyone learn a little computational thinking? Cuny, Snyder and I advocate these benefits [CunySnyderWing10]:

Computational thinking for everyone means being able to:

- Understand which aspects of a problem are amenable to computation,
- Evaluate the match between computational tools and techniques and a problem,
- Understand the limitations and power of computational tools and techniques,
- Apply or adapt a computational tool or technique to a new use,
- Recognize an opportunity to use computation in a new way, and
- Apply computational strategies such divide and conquer in any domain.

Computational thinking for scientists, engineers, and other professionals further means being able to:

- Apply new computational methods to their problems,
- Reformulate problems to be amenable to computational strategies,
- Discover new science through analysis of large data,
- Ask new questions that were not thought of or dared to ask because of scale, but which are easily addressed computationally, and
- Explain problems and solutions in computational terms.

Computational Thinking in Education

Campuses throughout the United States and abroad are revisiting their undergraduate curriculum in computer science. Many are changing their first course in computer science to cover fundamental principles and concepts, not just programming. For example, at Carnegie Mellon we recently revised our undergraduate first-year courses to promote computational thinking for non-majors [Link10].

Moreover, the interest and excitement surrounding computational thinking has grown beyond undergraduate education to additional recent projects, many focused on incorporating computational thinking into kindergarten through 12th grade education. Sponsors include professional organizations, government, academia and industry.

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 website at www.csprinciples.org). Five universities are piloting versions of this course this year: University of North Carolina at Charlotte, University of California at Berkeley, Metropolitan State College of Denver, University of California at San Diego and University of Washington. The plan is for more schools--high schools, community colleges and universities--to participate next year.

Computer science is also getting attention from elected officials. In May 2009, computer science thought leaders held an event on Capitol Hill to call on policymakers to put the "C" in STEM, that is, 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 event was sponsored by ACM, CRA, CSTA, IEEE, Microsoft, NCWIT, NSF, and SWE .

The U.S. House of Representatives has now designated the first week of December as Computer Science Education Week (www.csedweek.org); the event is sponsored by ABI, ACM, BHEF, CRA, CSTA, Dot Diva, Google, Globaloria, Intel, Microsoft, NCWIT, NSF, SAS, and Upsilon Pi Epsilon. In July 2010, U.S. Rep. Jared Polis (D-CO) introduced the Computer Science Education Act (H.R. 5929) in an attempt to boost K-12 computer science education efforts.

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 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 proposed 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 has also helped promote the vision 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." Those 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 2010, under the auspices of Google, CS4HS had spread to 20 schools in the United States and 14 in Europe, the Middle East and Africa. Also at Carnegie Mellon, Microsoft Research funds the Center for Computational Thinking (www.cs.cmu.edu/~CompThink/), which supports both research and educational outreach projects.

Computational thinking has also spread internationally. In August 2010, the Royal Society--the U.K.'s equivalent of the U.S.'s National Academy of Sciences--announced that it is leading an 18-month project to look "at the way that computing is taught in schools, with support from 24 organizations from across the computing community including learned societies, professional bodies, universities and industry." (See www.royalsociety.org/education-policy/projects/.) One organization that has already taken up the challenge in the U.K. is called Computing At School, a coalition run by the British Computing Society and supported by Microsoft Research and other industry partners.

Resources Abound

The growing worldwide focus on computational thinking means that resources are becoming available for educators, parents, students and everyone else interested in the topic.

In October 2010, Google launched the Exploring Computational Thinking website (www.google.com/edu/computational-thinking), which has a wealth of links to further web resources, including lesson plans for K-12 teachers in science and mathematics.

Computer Science Unplugged (www.csunplugged.org), created by Tim Bell, Mike Fellows and Ian Witten, teaches computer science without the use of a computer. It is especially appropriate for elementary and middle school children. Several dozen people working in many countries, including New Zealand, Sweden, Australia, China, Korea, Taiwan and Canada, as well as in the United States, contribute to this extremely popular website.

The National Academies' Computer Science and Telecommunications Board held a series of workshops on "Computational Thinking for Everyone" with a focus on identifying the fundamental concepts of computer science that can be taught to K-12 students. The first workshop report [NRC10] provides multiple perspectives on computational thinking.

Additionally, panels and discussions on computational thinking have been plentiful at venues such as the annual ACM Special Interest Group on Computer Science Education (SIGCSE) symposium and the ACM Educational Council. The education committee of the CRA presented

a white paper [CRA-E10] at the July 2010 CRA Snowbird conference, which includes recommendations for computational thinking courses for non-majors. CSTA produced and distributes "Computational Thinking Resource Set: A Problem-Solving Tool for Every Classroom." It's available for download at the CSTA's website (www.csta.acm.org).

Final Remarks--and a Challenge

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.

Computer scientists already know the value of thinking abstractly, thinking at multiple levels of abstraction, abstracting to manage complexity, abstracting to scale up, etc. Our immediate task ahead is to better explain to non-computer scientists what we mean by computational thinking and the benefits of being able to think computationally. Please join me in helping to spread the word!

Jeannette Wing is head of the Computer Science Department at Carnegie Mellon University and the President's Professor of Computer Science. She earned her bachelor's, master's and doctoral degrees at the Massachusetts Institute of Technology and has been a member of the Carnegie Mellon faculty since 1985.
From 2007 to 2010, Wing served as assistant director for the Computer and Information Science and Engineering Directorate of the National Science Foundation. She is a fellow of the American Academy of Arts and Sciences, the American Association for the Advancement of Science, the Association for Computing Machinery and the Institute of Electrical and Electronic Engineers.

Bibliography and Further Reading

[CRA-E10] Computer Research Association, "Creating Environments for Computational Researcher Education," August 9, 2010.
http://www.cra.org/uploads/documents/resources/rissues/CRA-E-Researcher-Education.pdf

[CunySnyderWing10] Jan Cuny, Larry Snyder and Jeannette M. Wing, "Demystifying Computational Thinking for Non-Computer Scientists," work in progress, 2010

[NRC10] "Report of a Workshop on the Scope and Nature of Computational Thinking," National Research Council, 2010 (www8.nationalacademies.org/cp/projectview.aspx?key=48969)

[Link10] Randal E. Bryant, Klaus Sutner and Mark Stehlik, "Introductory Computer Science Education: A Deans' Perspective," The Link, Fall 2010

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

[Wing06] Jeannette M. Wing, "Computational Thinking," Communications of the ACM, Vol.

49, No. 3, March 2006, pp. 33-35

Acronyms of Organizations:

ABI: Anita Borg Institute for Women and Technology
ACM: Association for Computing Machinery
BHEF: Business-Higher Education Forum
CISE: Computer and Information Science and Engineering
CRA: Computing Research Association
CRA-E: Computing Research Association-Education
CSTA: Computer Science Teachers Association
CSTB: Computer Science and Telecommunications Board
IEEE: Institute for Electrical and Electronic Engineers
NCWIT: National Center for Women and Information Technology
NSF: National Science Foundation
SIGCSE: ACM Special Interest Group on Computer Science Education
SWE: Society for Women Engineers

Acronyms of NSF Programs:

BPC: Broadening Participating in Computing
CDI: Cyber-enabled Discovery and Innovation
CE21: Computing Education for the 21st Century
CPATH: CISE Pathways to Revitalized Undergraduate Computing Education
For More Information:

Jason Togyer | 412-268-8721