A reprint from

American Scientist
the magazine of Sigma Xi, The Scientific Research Society

The Great Principles of Computing

Peter J. Denning

Computing is integral to science—not just as a tool for analyzing data, but as an agent of thought and discovery.

It has not always been this way. Computing is a relatively young discipline. It started as an academic field of study in the 1930s with a cluster of remarkable papers by Kurt Gödel, Alonzo Church, Emil Post and Alan Turing. The papers laid the mathematical foundations that would answer the question "what is computation?" and discussed schemes for its implementation. These men saw the importance of automatic computation and sought its precise mathematical foundation. The various schemes they each proposed for implementing computation were quickly found to be equivalent, as a computation in any one could be realized in any other. It is all the more remarkable that their models all led to the same conclusion that certain functions of practical interest—such as whether a computational algorithm (a method of evaluating a function) will ever come to completion instead of being stuck in an infinite loop—cannot be answered computationally.

Computing may be the fourth great domain of science along with the physical, life and social sciences

At the time that these papers were written, the terms "computation" and "computers" were already in common use, but with different connotations from today. Computation was taken to mean the mechanical steps followed to evaluate mathematical functions; computers were people who did computations. In recognition of the social changes they were ushering in, the designers of the first digital computer projects all named their systems with acronyms ending in "-AC", meaning automatic computer—resulting in names such as ENIAC, UNIVAC and EDSAC.

At the start of World War II, the militaries of the United States and the United Kingdom became interested in applying computation to the calculation of ballistic and navigation tables and to the cracking of ciphers. They commissioned projects to design and build electronic digital computers. Only one of the projects was completed before the war was over. That was the top-secret project at Bletchley Park in England, which cracked the German Enigma cipher using methods designed by Alan Turing.

Many people involved in those projects went on to start computer companies in the early 1950s. Universities began offering programs of study in the new field in the late 1950s. The field and the industry have grown steadily into a modern behemoth whose Internet data centers are said to consume almost three percent of the world's electricity.

During its youth, computing was an enigma to the established fields of science and engineering. At first, computing looked like only the applied technology of math, electrical engineering or science, depending on the observer. However, over the years, computing provided a seemingly unending stream of new insights, and it defied many early predictions by resisting absorption back into the fields of its roots. By 1980 computing had mastered algorithms, data structures, numerical methods, programming languages, operating systems, networks, databases, graphics, artificial intelligence and software engineering. Its great technological achievements—the chip, the personal computer and the Internet—brought it into many lives. These advances stimulated more new subfields, including network science, Web science, mobile computing, enterprise computing, cooperative work, cyberspace protection, user-interface design and information visualization. The resulting commercial applications have spawned

Peter J. Denning is Director of the Cebrowski Institute for Innovation and Information Superiority at the Naval Postgraduate School in Monterey, California, and is a past president of ACM. Email: pjd@nps.edu

Category	Focus	Examples
Computation	What can and cannot be computed	Classifying complexity of problems in terms of the number of computational steps to achieve a solution
Communication	Reliably moving information between locations	Information measured as entropy. Compression of files, error-correcting codes, cryptography
Coordination	Effectively using many autonomous computers	Protocols that eliminate conditions that cause indeterminate results
Recollection	Representing, storing, and retrieving information from media	All storage systems are hierarchical, but no storage system can offer equal access time to all objects. All computations favor subsets of their data objects in any time interval
Automation	Discovering algorithms for information processes	Most heuristic algorithms can be formulated as searches over enormous data spaces. Many human cognitive processes can be modeled as information processes
Evaluation	Predicting performance of complex systems	Most computational systems can be modeled as networks of servers whose fast solutions yield close approximations of real throughput and response time
Design	Structuring software systems for reliability and dependability	Complex systems can be decomposed into interacting modules and virtual machines. Modules can be stratified corresponding to their time scales of events that manipulate objects

The Great Principles of Computing framework is designed to give a scientific definition of the field. The principles fall into seven categories, each of which is defined and given examples above.

new research challenges in social networks, endlessly evolving computation, music, video, digital photography, vision, massive multiplayer online games, user-generated content and much more.

The name of the field has changed several times to keep up with the flux. In the 1940s it was called *automatic computation* and in the 1950s, *information processing*. In the 1960s, as it moved into academia, it acquired the name *computer science* in the U.S. and *informatics* in Europe. By the 1980s computing comprised a complex of related fields, including computer science, informatics, computational science, computer engineering, software engineering, information systems and information technology. By 1990 the term *computing* had become the standard for referring to this core group of disciplines.

Computing's Paradigm

Traditional scientists frequently questioned the name *computer science*. They could easily see an engineering paradigm (design and implementation of systems) and a mathematics paradigm (proofs of theorems) but they could not see much of a science paradigm (experimental verification of hypotheses). Moreover, they understood science as a way of dealing with the natural world, and computers looked suspiciously artificial.

The founders of the field came from all three paradigms. Some thought computing was a branch of applied mathematics, some a branch of electrical engineering, and some a branch of computational-oriented science. During its first four decades, the field focused primarily on engineering: The challenges of building reliable computers, networks and complex software were daunting and occupied almost everyone's attention. By the 1980s these challenges largely had been met and computing was spreading rapidly into all fields, with the help of networks, supercomputers and personal computers. During the 1980s computers became powerful enough that science visionaries could see how to use them to tackle the hardest questions—the "grand challenge" problems in science and engineering. The resulting "computational science" movement involved scientists from all countries and culminated in the U.S. Congress's adoption of the High-Performance Computing and Communications (HPCC) Act of 1991 to support research on a host of large problems.

Today, there is an agreement that computing *exemplifies* science and engineering, and that neither science nor engineering *characterizes* computing. Then what does? What is computing's paradigm?

The leaders of the field struggled with this paradigm question from the beginning. Along the way, there were three waves of attempts to unify views. Allen Newell, Alan Perlis and Herb Simon led the first one in 1967. They argued that computing was unique among all the sciences in its study of information processes. Simon, a Nobel laureate in economics, went so far as to call computing a science of the artificial. A catchphrase of this wave was "computing is the study of phenomena surrounding computers."

The second wave focused on programming, the art of designing algorithms that produce information processes. In the early 1970s, computing pioneers Edsger Dijkstra and Donald Knuth took strong stands favoring algorithm analysis as the unifying theme. A catchphrase of this wave was "computer science equals programming." In recent times, this view has foundered because the field has expanded well beyond programming, whereas the public understanding of a programmer has narrowed to just those who write code.

The third wave came as a result of the Computer Science and Engineering Research Study (COSERS), led by Bruce Arden in the late 1970s. Its catchphrase was "computing is the automation of information processes." Although its final report successfully exposed the science in computing and explained many esoteric aspects to the layperson, its central view did not catch on.

An important aspect of all three definitions was the positioning of the computer as the object of attention. The computational-science movement of the 1980s began to step away from that notion, adopting the view that computing is not only a tool for science, but also a new method of thought and discovery in science. The process of dissociating from the computer as the focal point came to completion in the late 1990s when leaders in the field of biology—epitomized by Nobel laureate David Baltimore and echoing cognitive scientist Douglas Hofstadter—said

	Physical	Social	Life	Computing
Computing implemented by:	mechanical, optical, electronic, quantum and chemical computing	mechanical robots, human cognition, games with inputs and outputs	genomic, neural, immunological, DNA translation, evolutionary computing	compilers, operating systems, emulation, abstractions, procedures, architectures, languages
Computing implements:	modeling, simulation, databases, data systems, quantum cryptography	artificial intelligence, cognitive modeling, autonomic systems	artificial life, biomimetics, systems biology	
Computing influenced by:	sensors, scanners, computer vision, optical character recognition, localization	learning, programming, user modeling, authorization, speech understanding	eye, gesture, expression, and movement tracking; biosensors	networking, security, parallel computing, distributed systems, grids
Computing imfluences:	locomotion, fabrication, manipulation, open-loop control	screens, printers, graphics, speech generation, network science	bioeffectors, haptics, sensory immersion	
Bidirectional influence	robots, closed-loop control	human-computer interaction, games	brain-computer interfaces	

Computing interacts in many ways with the other domains of science. Computing implements a phenomenon by generating its behaviors. Examples of how computing is both implemented by, and implements, the domains of physics, social and life sciences, and well as influencing its own behaviors, are given above.

that biology had become an information science and DNA translation is a natural information process. Many computer scientists have joined biologists in research to understand the nature of DNA information processes and to discover what algorithms might govern them.

Take a moment to savor this distinction that biology makes. First, some information processes are natural. Second, we do not know whether all natural information processes are produced by algorithms. The second statement challenges the traditional view that algorithms (and programming) are at the heart of computing. Information processes may be more fundamental than algorithms.

Scientists in other fields have come to similar conclusions. They include physicists working with quantum computation and quantum cryptography, chemists working with materials, economists working with economic systems, and social scientists working with networks. They have all said that they have discovered information processes in their disciplines' deep structures. Stephen Wolfram, a physicist and creator of the software program *Mathematica*, went further, arguing that information processes underlie every natural process in the universe.

All this leads us to the modern catchphrase: "Computing is the study of information processes, natural and artificial." The computer is a tool in these studies but is not the object of study. As Dijkstra once said, "Computing is no more about computers than astronomy is about telescopes."

The term *computational thinking* has become popular to refer to the mode of thought that accompanies design and discovery done with computation. This term was originally called *algorithmic thinking* in the 1960s by Newell, Perlis and Simon, and was widely used in the 1980s as part of the rationale for computational science. To think computationally is to interpret a problem as an information process and then seek to discover an algorithmic solution. It is a very powerful paradigm that has led to several Nobel Prizes.

Great Principles of Computing

The maturing of our interpretation of computing has given us a new view of the content of the field. Until the 1990s, most computing scientists would have said that it is about algorithms, data structures, numerical methods, programming languages, operating systems, networks, databases, graphics, artificial intelligence and software engineering. This definition is a technological interpretation of the field. A scientific interpretation would emphasize the fundamental principles that empower and constrain the technologies.

My colleagues and I have developed the Great Principles of Computing framework to accomplish this goal. These principles fall into seven categories: computation, communication, coordination, recollection, automation, evaluation and design *(see the first table for examples).*

Each category is a perspective on computing, a window into the knowledge space of computing. The categories are not mutually exclusive. For example, the Internet can be seen as a communication system, a coordination system or a storage system. We have found that most computing technologies use principles from all seven categories. Each category has its own weight in the mixture, but they are all there.

In addition to the principles, which are relatively static, we need to take account of the dynamics of interactions between computing and other fields. Scientific phenomena can affect one another in two ways: implementation and influence. A combination of existing things implements a phenomenon by generating its behaviors. Thus, digital hardware physically implements computation; artificial intelligence implements aspects of human thought; a compiler implements a high-level language with machine code; hydrogen and oxygen implement water; complex combinations of amino acids implement life.

Influence occurs when two phenomena interact with each other. Atoms arise from the interactions among the forces generated by protons, neutrons and electrons. Galaxies interact via gravitational waves. Humans interact with speech, touch and computers. And interactions exist across domains as well as within domains. For example, computation influences physical action (electronic controls), life processes (DNA translation) and social processes (games with outputs). The second table illustrates interactions between computing and each of the physical, life and social sciences, as well as within computing itself. There can be no question about the pervasiveness of computing in all fields of science.

What Are Information Processes?

There is a potential difficulty with defining computation in terms of information.

Information seems to have no settled definition. Claude Shannon, the father of information theory, in 1948 defined information as the expected number of yes-or-no questions one must ask to decide what message was sent by a source. He purposely skirted the issue of the meaning of bit patterns, which seems to be important to defining information. In sifting through many published definitions, Paolo Rocchi in 2010 concluded that definitions of information necessarily involve an objective component—signs and their referents, or in other words, symbols and what they stand for—and a subjective component—meanings. How can we base a scientific definition of information on something with such an essential subjective component?

Biologists have a similar problem with "life." Life scientist Robert Hazen notes that biologists have no precise definition of life, but they do have a list of seven criteria for when an entity is living. The observable affects of life, such as chemistry, energy and reproduction, are sufficient to ground the science of biology. In the same way, we can ground a science of information on the observable affects (signs and referents) without having a precise definition of meaning.

A representation is a pattern of symbols that stands for something. The association between a representation and what it stands for can be recorded as a link in a table or database, or as a memory in people's brains. There are two important aspects of representations: *syntax* and *stuff*. Syntax is the rules for constructing patterns; it allows us to distinguish patterns that stand for something from patterns that do not. Stuff is the measurable physical states of the world that hold representations, usually in media or signals. Put these two together and we can build machines that can detect when a valid pattern is present.

A representation that stands for a method of evaluating a function is called an algorithm. A representation that stands for values is called data. When implemented by a machine, an algorithm controls the transformation of an input data representation to an output data representation. The algorithm representation controls the transformation of data representations. The distinction between the algorithm and the data representations is pretty weak; the executable code generated by a compiler looks like data to the compiler and like an algorithm to the person running the code.

Even this simple notion of representation has deep consequences. For example, as Gregory Chaitin has shown, there is no algorithm for finding the shortest possible representation of something.

Some scientists leave open the question of whether an observed information process is actually controlled by an algorithm. DNA translation can thus be called an information process; if someone discovers a controlling algorithm, it could be also called a computation.

Some mathematicians define computation as separate from implementation. They treat computations as logical orderings of strings in abstract languages, and are able to determine the logical limits of computation. However, to answer questions about the running time of observable computations, they have to introduce costs—the time or energy of storing, retrieving or converting representations. Many real-world problems require exponential-time computations as a consequence of these implementable representations. My colleagues and I still prefer to deal with implementable representations because they are the basis of a scientific approach to computation.

These notions of representations are sufficient to give us the definitions we need for computing. An information process is a sequence of representations. (In the physical world, it is a continuously evolving, changing representation.) A computation is an information process in which the transitions from one element of the sequence to the next are controlled by a representation. (In the physical world, we would say that each infinitesimal time and space step is controlled by a representation.)

Where Computing Stands

Computing as a field has come to exemplify good science as well as engineering. The science is essential to the advancement of the field because many systems are so complex that experimental methods are the only way to make discoveries and understand limits. Computing is now seen as a broad field that studies information processes, natural and artificial.

This definition is wide enough to accommodate three issues that have nagged computing scientists for many years: Continuous information processes (such as signals in communication systems or analog computers), interactive processes (such as ongoing Web services) and natural processes (such as DNA translation) all seemed like computation but did not fit the traditional algorithmic definitions.

The great-principles framework reveals a rich set of rules on which all computation is based. These principles interact with the domains of the physical, life and social sciences, as well as with computing technology itself.

Computing is not a subset of other sciences. None of those domains are fundamentally concerned with the nature of information processes and their transformations. Yet this knowledge is now essential in all the other domains of science. Computer scientist Paul Rosenbloom of the University of Southern California in 2009 argued that computing is a new great domain of science. He is on to something.

Bibliography

Arden, B. W., ed. 1983. *What Can Be Automated: Computer Science and Engineering Research Study (COSERS).* Cambridge, MA: The MIT Press.

Bacon, D., and W. van Dam. 2010. Recent progress in quantum algorithms. *Communications of the ACM* 53:84–93.

Baltimore, D. 2001. How Biology Became an Information Science. In *The Invisible Future,* P. Denning, ed. New York, NY: McGraw-Hill.

Chaitin, G. 2006. *Meta Math! The Quest for Omega.* New York, NY: Vintage Press.

Denning, P. 2003. Great Principles of Computing. *Communications of the ACM* 46:15–20.

Denning, P., and C. Martell. Great Principles of Computing Website. http://greatprinciples.org

Denning, P. 2007. Computing is a natural science. *Communications of the ACM* 50:15–18.

Denning, P., and P. Freeman. 2009. Computing's paradigm. *Communications of the ACM* 52: 28–30.

Hazen, R. 2007. Genesis: *The Scientific Quest for Life's Origins.* Washington, D.C.: Joseph Henry Press.

Hofstadter, D. 1985. *Metamagical Themas: Questing for the Essence of Mind and Pattern.* New York, NY: Basic Books.

Newell, A., A. J. Perlis and H. A. Simon. 1967. Computer science. *Science* 157:1373–1374.

Rocchi, P. 2010. *Logic of Analog and Digital Machines.* Hauppauge, NY: Nova Publishers.

Rosenbloom, P. S. 2004. A new framework for computer science and engineering. *IEEE Computer* 31–36.

Shannon, C., and W. Weaver. 1949. *The Mathematical Theory of Communication.* Champaign, IL: University of Illinois Press. Available at http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html

Simon, H. 1969. *The Sciences of the Artificial.* Cambridge, MA: The MIT Press.

Wolfram, S. 2002. *A New Kind of Science.* Champaign, IL: Wolfram Media.