GREAT PRINCIPLES OF COMPUTING

Peter J. Denning, Naval Postgraduate School, Monterey, California

April 2008
(Rev. 8/31/08)

Abstract: The Great Principles of Computing is a framework for understanding computing as a field of science.

Keywords: computer science structure, body of knowledge, computation, communication, coordination, recollection, automation, evaluation, design

The Great Principles of Computing is a framework for understanding fundamental principles computing as an integrated field of science and engineering.

Few views of the computing field see the integral whole. Many outsiders see computing as a field of technology, gadgetry, and programming. Insiders often debate a separatist perspective -- whether computer science is mathematics, engineering, or science. Indeed, some skeptics even question whether the word "science" belongs at all -- to them "computer science" is a misnomer because the subject matter of the field is manmade artifacts, not natural objects.

It is a common practice in science to articulate scientific fields as frameworks of fundamental principles. Robert Hazen and James Trefil have done this for the biological and life sciences (1), Richard Feynman for physics (2), and Carl Sagan for Astronomy (3). Some computer scientists have taken steps in this direction, notably Alan Biermann (4) and Danny Hillis (5). A sea change of attitude among other fields of science toward computing, which started around 2000, makes it important to carry these steps forward to a complete fundamentals framework for computing.

By the mid 1990s, it became clear that many fields of science were discovering natural information processes in their deep structures. Biologists consider DNA transcription as an information process that decodes DNA and produces new cells. Physicists consider quantum waves as carriers of information that translate into particles and interactions; from this came quantum computing and quantum cryptography. Economists consider the workings of economies as complex information processes. Materials scientists consider molecules as objects that could be designed by manipulating energy in accordance with the Schrödinger equation. These fields and many more sought collaborations with computer scientists to help them understand the information processes they had discovered.

As we face the modern challenges of understanding nature's ways of computation, we find ourselves revisiting old, fundamental questions of computer science:

What is computation?
What is information?
What can we know through computing?
What can we not know through computing?
How can we build complex systems simply?

Computer scientists have studied these questions since the 1930s. Today, people in all fields of science, engineering, business, and even politics are asking the same questions. Even if it seems they are unanswerable, just engaging with them will advance the scientific and engineering foundations of computing.

Our tradition defines computer science as the study of phenomena surrounding computers. This definition is no longer workable because we are studying natural information processes as well as artificial. We are seeing that the computer is the tool and that computation is the principle. Computing is -- in fact, always has been -- the science and application of information processes, natural and artificial.

Evolution of Computing Frameworks

Although computer science was originally conceived as the study of computational processes (6), the practical challenges of building fast and reliable computers soon transferred the focus to the computers themselves. From around 1940, we described computing by the ideas in its core technologies -- such as logic circuits, algorithms, languages, programs, compilers, or operating systems (4). The computer was a tool for solving equations, cracking codes, analyzing data, and managing business processes. Computation was the activity of computers.

Over the next four decades, computing technology advanced and matured and the supercomputer wrought significant advances in science and engineering. By 1980, our understanding of computation had been shaken up. Computation was no longer just the activity of computers; it was a new method in science. It became the third leg of science along with theory and experiment. It also became a new method in engineering design.

During the 1990s, our understanding of computation was shaken up again. People in many fields discovered information processes in their deepest structures -- for example, DNA in biology, quantum waves in physics, brain patterns in cognitive science, and information flows in economic systems. Computation entered everyday life with new ways to solve problems, new forms of art, music, motion pictures, and commerce, new approaches to learning, new slang expressions, and even new political jokes ("What did Bill Clinton play on his saxophone? Al Gore rhythms.").

Thus the fundamental questions of computing, listed earlier, have become important in many fields, which now rely heavily on computation and

computational methods to advance their work (7-9). Not only that, but discoveries in other fields are yielding new fundamental computing principles.

The technology-based framework for computing, which served well for 50 years, came under stress in the 1990s because of its sheer complexity. In 1989, the ACM listed 9 core technologies of computing (10). In 2001, however, the ACM listed 14 core areas and 63 core topics under those areas (11). For the newcomer, learning the inner workings of all technologies and their possible direct interactions is a daunting challenge.

The Great Principles framework discussed below is much simpler. It has seven categories of principles with five to eight principles in each. Everything else flows from those base principles.

Is Computer Science Science?

Over the years, skeptics have asked whether "science" belongs in the title of the field. They have said that any field naming itself as a science cannot be. To them, computer science looks not like a science but a field of artifacts and concepts about them.

It is worth verifying that computing is really a field of science and, therefore, that a framework inspired by science is useful and meaningful. To be accepted as a science, a field of study must satisfy six criteria:

1. Systematically organized body of knowledge
2. An experimental method
3. Reproducible experimental results
4. Testable, falsifiable hypotheses
5. Surprising predictions
6. Natural objects

In 2005, we analyzed the first five criteria and concluded that computer science meets them all (12). However, skeptics still felt that computer science is an artificial science rather than a natural science. In 2007, we analyzed the sixth criterion and demonstrated that computing is a natural science (13).

General Considerations for a Framework

By a principle, we mean a statement that guides or constrains future action. Computing principles are of two kinds: (1) recurrences, including laws, processes, and methods that describe repeatable cause-effect relationships, and (2) guidelines for conduct. An example of a law is as follows: "The fastest sorting algorithms take time of order of n log n to arrange n items in order." An example of a conduct guideline is as follows: "Network designers should divide protocol software into layers." The purpose of such guidelines is to reduce apparent complexity, increase understanding, and enable good design.

By a framework of a field, we mean a set of principles statements and stories organized into categories (a taxonomy), accompanied by a rationale of how they fit together into a coherent body and how they influence technology. In

developing a framework for computing, we emphasized the deepest principles and called the result the Great Principles framework.

The benefits of developing and maintaining a Great Principles framework for computing are as follows:

- Stimulate deep thinking. Returning to the fundamental questions advances the field even if we never fully settle them.
- Expose deep structure. Doing so can reduce the apparent complexity of the field, contributing to greater understanding, better designs, and simpler, more reliable systems.
- Enable designers and users to see connections among technologies based on similar principles. This will facilitate sound designs, cross fertilization among technologies, new discoveries, and innovations.
- Establish a new relationship with people from other fields by offering computing principles in language that shows them how to map computing principles into their own fields.
- Provide inspiring stories about the development of the field and its principles for young people.
- Develop new approaches to teaching computing that inspire curiosity and excitement.

A Great Principles framework complements the existing technology frameworks for understanding computing. We will discuss this further below.

As with any other body of knowledge, a Great Principles framework evolves as new principles are discovered and old principles become obsolete. Examples of new principles include searching through very large distributed databases, avoiding information overload, and forming new networks rapidly. Examples of obsolete principles include construction of logic gates from discrete transistors, LR parsing, and centralized network routing tables. A principles framework is a living depiction of the field, always open to births and retirements. Its rate of change is much slower than for a technology-oriented depiction of the field.

Outline of a Framework

An examination of many computing technologies for their foundational principles led to a framework of seven categories (14):

Computation (meaning and limits of computing)

Communication (reliable data transmission)

Coordination (networked entities working toward common goals)

Recollection (storage and retrieval of information)

Automation (meaning and limits of automation)

Evaluation (performance prediction and capacity planning)

Design (building reliable computing systems)

These categories resulted from a functional analysis of many computing technologies and applications:

1. Computing systems are built from processing elements that transform and store information (computation, recollection);
2. Processing elements exchange information (communication);
3. Processing elements cooperate toward common goals (coordination);
4. Humans delegate tasks to systems of processing elements (automation);
5. Humans predict the speed and capacity of systems (evaluation); and
6. Humans decompose systems into processing elements and organize their construction (design).

The seven categories are like windows into the one computing knowledge space rather than slices of the space into separate pieces (Fig. 1). Each window sees the space in a distinctive way, but the same thing can be seen in more than one window. Internet protocols, for example, are seen variously as means for data communication, coordination, and recollection.

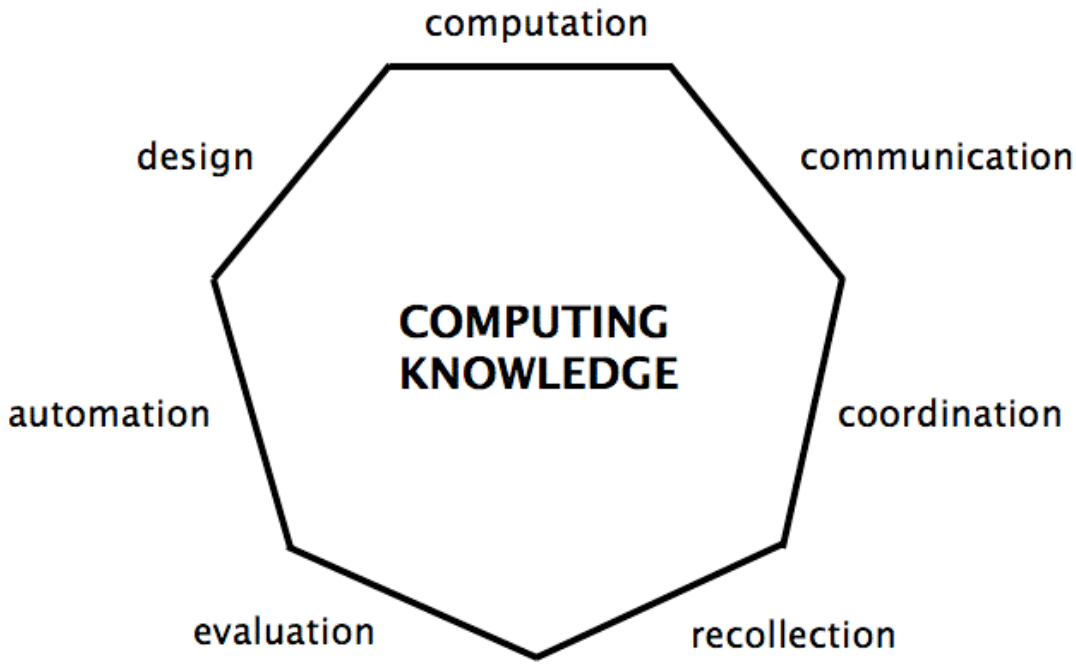

Figure 1. Categories of the Great Principles (GP) framework.

Moreover, most computing technologies draw principles from all seven categories. A principles framework exposes many common factors among technologies.

A Summary of Principles

The list that follows gives key principles in each of the seven categories. This is intended not as a "final" statement of principles, but a starting point for fleshing out a framework. In time, omissions will be overcome, statements clarified, and the number of statements reduced. The intention is that the operations of all the technologies linked to each category can be derived from one or more of the stated principles.

Computation

- Representations hold information.
- Computation is a sequence of representations.
- Computations can be open or closed.
- Computations have characteristic speeds of resolution.
- Complexity measures the time or space essential to complete computations.
- Finite representations of real processes always contain errors.

Communication

- Information can be encoded into messages.
- Data communication always takes place in a system consisting of a message source, an encoder, a channel, and a decoder.
- Information in a message source places a hard lower bound on channel capacity for accurate reception (Shannon Capacity Theorem).
- Messages corrupted during transmission can be recovered during reception (Error Correction).
- Messages can be compressed.
- Messages can hide information.

Coordination

- A coordination system is a set of agents interacting within a finite or infinite game toward a common objective.
- Action loop is the foundational element of all coordination protocols.
- Coordination tasks can be delegated to computational processes.
- The protocols of coordination systems manage dependencies of flow, sharing, and fit among activities.
- It is impossible to select one of several simultaneous or equally attractive alternatives within a preset deadline (Choice Uncertainty Principle).

- All coordination systems depend on solutions to the concurrency control problems of arbitration, synchronization, serialization, determinacy, and deadlock.

Recollection

- All computations take place in storage systems.
- Storage systems comprise hierarchies with volatile (fast) storage at the top and persistent (slower) storage at the bottom.
- The principle of locality dynamically identifies the most useful data, which can be cached at the top of the hierarchy.
- Thrashing is a severe performance degradation caused when parallel computations overload the storage system.
- Access to stored objects is controlled by dynamic bindings among names, handles, addresses, and locations.
- Hierarchical naming systems allow local authorities to assign names that are globally unique in very large name spaces.
- Handles enable sharing by providing unique-for-all-time object identifiers that are independent of all address spaces.
- Data can be retrieved by name or by content.

Automation

- Physical automation maps hard computational tasks to physical systems that perform them acceptably well.
- Artificial intelligence maps human cognitive tasks to physical systems that perform them acceptably well.
- Artificial intelligence maps tasks to systems through models, search, deduction, induction, and collective intelligence.
- Models represent processes by which intelligent beings generate their behavior.
- Search finds the subsets of states of a complex system that must participate in the final outcome of a task.
- Deduction locates the outcome of a task by applying rules of logic to move from axioms to provable statements.
- Induction builds models by generalizing from data about a complex task's behavior.
- Collective intelligence exploits large-scale aggregation and coordination in networks to produce new knowledge.

Evaluation

- The principal tools of evaluation are modeling, simulation, experiment, and statistical analysis of data.
- Computing systems can be represented as sets of equations balancing transition flows among states.
- Network of servers is a common, efficient representation of computing systems.
- Network-of-server systems obey fundamental laws on their utilizations, throughputs, queueing, response times, and bottlenecks.
- Resource sharing, when feasible, is always more efficient than partitioning.

Design

- Design principles are conventions for planning and building correct, fast, fault-tolerant, and fit computing systems.
- Error confinement and recovery are much harder in the virtual worlds of computing than in the real world of physical objects.
- The four base principles of computer system design are hierarchical aggregation, levels, virtual machines, and objects.
- Abstraction, information hiding, and decomposition are complementary aspects of modularity.
- Levels organize the functions of a system into hierarchies that allow downward invocations and upward replies.
- Virtual machines organize software as simulations of computing machines.
- Objects organize software into networks of shared entities that activate operations in each other by exchanging signals.
- In a distributed system, it is more efficient to implement a function in the communicating applications than in the network itself (end-to-end principle).

Principle-Stories

The list above is a set of statements. Simple statements, however, do not capture the richness or the surprising implications of a principle. For this reason many fields express their principles with stories. In physics, for example, the main terms -- such as photons, electrons, quarks, quantum wave function, relativity, and energy conservation -- are actually the titles of stories. So it is in astronomy for planets, stars, galaxies, quasars, black holes, and Hubble shift; in thermodynamics for entropy, first law, second law, and Carnot cycle; in biology for phylogeny, ontogeny, DNA, and enzymes; and in electrical engineering for vacuum tube amplification, oscillator feedback, AM-FM radio, and transistors.

The principles of a field are actually a set of interwoven stories about the structure and behavior of field elements. They are the names of chapters in books about the field.

Principle-stories seek to make simple the complex history of a complex area. They tell how the principle evolved and grew in acceptance over time. They name the main contributors. They chronicle feats of heroes and failures of knaves. They lay out obstructions and how they were overcome. They explain how the principle works and how it affects everything else. The game is to define many terms in terms of a few terms and to derive many statements logically from a few statements.

Principle stories for computing are still relatively uncommon. We have many stories tracing inventions and innovations but not very many tracing the discovery of a principle and its effects on practice. Such stories will become more common in the future.

Using a Principles Oriented Body of Knowledge

A body of knowledge (BOK) is an organized description of the knowledge of a field. There are two basic, useful strategies for representing a field's BOK: enumerate its technologies, and enumerate its principles. They are different interpretations of the same knowledge space. The different kinds of users -- particularly the philosopher, the interested outsider, the technology designer, and the educator -- can act in new, useful ways with a principles representation.

Curriculum developers often work with a BOK so that they can be sure that they cover the essential knowledge of their field. The ACM includes a computing BOK in its Curriculum 2001 recommendation (11). That BOK is a list of core technologies of the computing field, the ones every computing professional should know. The main headings of the ACM BOK are core technologies, methods, devices, processes, or key concepts.

A principles framework is orthogonal to a technology-oriented framework. The same principle may appear in several technologies, and any technology relies on several principles. The set of active principles (those used in at least one technology) evolves much more slowly than the technologies.

Although the two styles of framework are different, they are strongly connected. To see the connection, imagine a two-dimension matrix. The rows are the topics from a technology-oriented framework, and the columns are the categories of principles. (We sometimes refer to the categories as windows because each category represents a distinctive way of viewing technologies.) The interior of the matrix is the knowledge space of the field (Fig. 2).

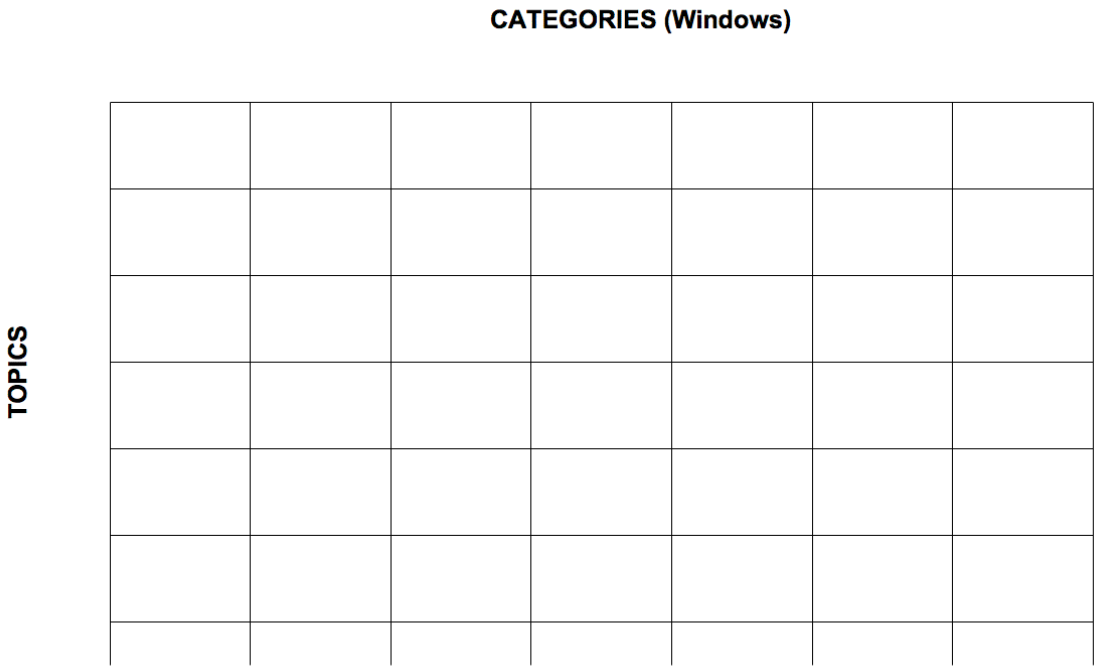

Figure 2. Computing knowledge space as a 2-D structure of technology topics and principles categories.

Under topics, we can list the technologies from the ACM BOK. Under categories, we can list the seven windows of the GP framework. A box in the matrix can be inscribed with the principles from the category (column) expressed through the technology (row). Figure 3 illustrates a matrix and a few of the many technology names; security technology shows two principles in the coordination category.

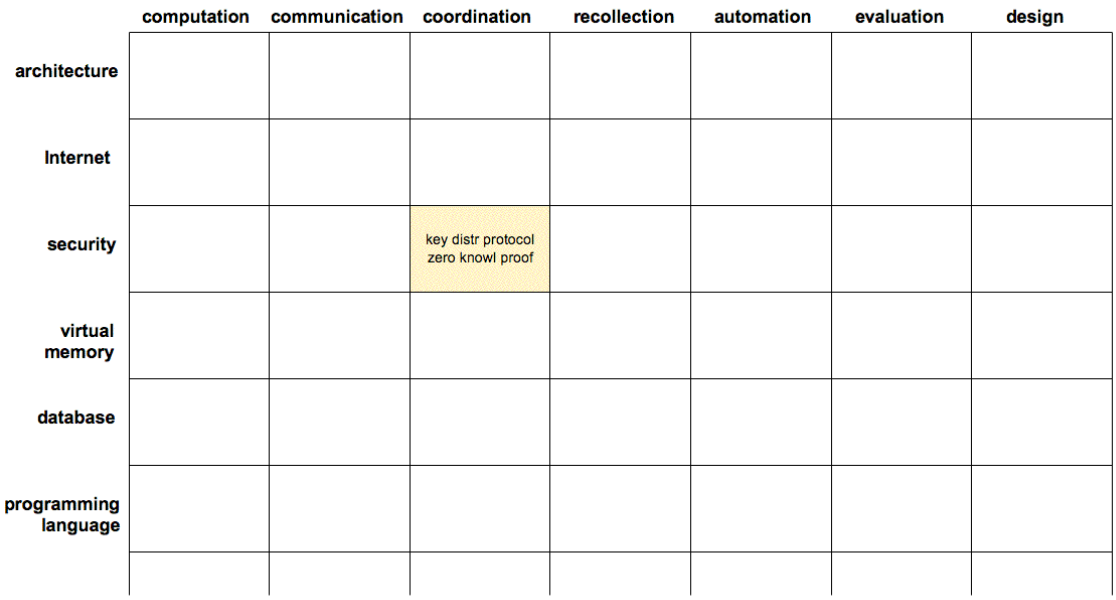

Figure 3. A portion of knowledge space covering six technology topics. Two coordination principles for security technology are shown (key distribution protocols and zero knowledge proofs).

Thus, the technology-oriented BOK enumerates the knowledge by rows of the matrix, whereas the principles-oriented BOK enumerates by columns. The important point is that they see the same knowledge -- from different perspectives and interpretations.

To illustrate this further, imagine someone who wants to enumerate all the principles involved with a technology. The answer is obtained simply by analyzing the technology for its principles in each of the seven categories. In Fig. 4, the security topic draws principles from all seven categories.

	computation	communication	coordination	recollection	automation	evaluation	design
security	O(.) of encryption functions	secrecy authentication covert channels	key distr protocol zero knowl proof	confinement partitioning for MLS reference monitor	intrusion detection biometric id	protocol perform under various loads	end-to-end layered functions virtual machines

Figure 4. Security technology draws principles from all seven categories.

The Great Principles framework opens new kinds of questions. For example, someone can enumerate all the technologies that employ a particular principle, or category of principles, as indicated in Figure 5, which shows an enumeration of security coordination principles. We see that the coordination category contributes principles to all the technologies listed.

	computation	communication	coordination	recollection	automation	evaluation	design
architecture			hardware handshake				
Internet			TCP and IP protocols				
security			key distr protocol zero knowl proof				
virtual memory			page fault interrupt				
database			locking protocol				
programming language			semaphores monitors				

Figure 5. Coordination principles are key parts of the six technologies.

Among the many likely users for a Great Principles framework, four groups stand out:

Philosophers of Science. These people inquire into the deep structures and truths of the field. They seek to understand what is true before science and engineering "facts" are established. They raise the bar for reflection and interpretation in the field.

Interested outside observers. These are people in other fields, or newcomers to the computing field, who seek a conceptual road map for the field. With a principles framework, they can explore the breadth and scope of the field, learn about the powers and limitations of all computing technologies, and find out how various aspects of computation work.

Technology designers. These are people who design or refine technologies, including innovators who seek new technologies. With a principles framework, they can

1. See the guiding principles of a technology;
2. See the limitations of a technology (imposed by principles);
3. See ways to simplify and reduce the perceived complexity of a technology; and
4. Find connections with other technologies that share the same principles.

Educators. These are people who teach computing and design curricula. With a principles framework, they can

1. Check the completeness of a BOK;
2. Given a topic for a course, determine the principles that must be brought out in the course;
3. Given a principle, find examples of technologies that exemplify it;
4. Design a course or series of lectures around a principle and the technologies that exploit it; and
5. Design a science map of the field.

Science and Art in Computing

To help define the boundaries of a field, we distinguish science from art. Art refers to the useful practices of a field (not to drawings or sculptures). Table 1 lists some terms that are often associated with science and with art. Programming, design, software and hardware engineering, building and validating models, and building user interfaces are all examples of "computing arts". If aesthetics is added, the computing arts extend to graphics, layout, drawings, photography, animation, music, games, and entertainment. All this computing art complements and enriches the science. Don Knuth once said that everything in the field begins at art and becomes science only when we understand it really well (15) .

Science versus Art

Science	Art
principles	practice
fundamental recurrences	skilled performance
theory	engineering
explanation	action
discovery	invention
analysis	synthesis
dissection	construction

To be complete, therefore, the framework needs to provide an account of computing practice as well as computing principles. Our competence is judged not by our ability to explain principles but by the quality of our performance. There are four main categories of computing practice:

- ***Programming --*** Using programming languages to build software systems that meet specifications created in cooperation with the users of those systems. Computing professionals must be multilingual and facile with the numerous programming languages (each attuned to a set of problem-solving strategies).
- ***Engineering Systems --*** Designing and constructing systems of software and hardware components running on servers connected by networks. These practices include an architectural design component concerned with organizing a system to produce valuable and tangible benefits for the users; an engineering component concerned with the modules, abstractions, revisions, design decisions, and risks in the system; and an operations component concerned with configuration, management, and maintenance of the system. High levels of skill are needed for large programmed systems encompassing thousands of modules and millions of lines of code.
- ***Modeling and Validation --*** Building models of natural and artificial systems to make predictions about their behavior under various conditions; and designing experiments to validate algorithms and systems.
- ***Innovating --*** Exercising leadership to design and bring about lasting changes to the ways groups and communities operate. Innovators watch for and analyze opportunities, listen to customers, formulate offers customers see as valuable, manage commitments to deliver the promised results, and inspire adoption. Innovators have strong historical sensibilities.

Summary

The Great Principles framework articulates the principles of the science and engineering of computing. We can view the field through seven themes -- computation, communication, coordination, recollection, automation, evaluation, and design -- that are interwoven into a single tapestry called computing. The principles listed here are a "proof of concept" that a compelling principles framework can be built. However, it will take some years of community discussion before everyone accepts a "final" framework.

The framework offers an integrated view of the field. Mathematics, engineering, and science appear in every category and often within the same principle. It is no longer necessary to ask is CS mathematics, engineering, or science. Computing stands on its own.

A Great Principles framework complements, but does not replace, technology frameworks for the field. It deals with the reality that computing is a natural science as well as a field of engineering. The framework gives a common language for computing that facilitates collaboration between computing people and other fields.

We hope that this framework invites the reader to think "out-of-the-box" and to understand computing in a broader context that serves many fields.

References

1. R. Hazen and J. Trefil, *Science Matters,* Sioux City, IA: Anchor, 1991.
2. R. Feynman, *Lectures in Physics,* New York: Addison-Wesley, 1970.
3. C. Sagan, *Cosmos,* New York: Random House, 2002.
4. A. Biermann, *Great Ideas in Computer Science* 2nd ed., Cambridge, MA: MIT Press, 1997.
5. D. Hillis, *The Pattern on the Stone,* Jackson, TN: Basic Books, 1999.
6. A. M. Turing, On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc., 2-42* 1936, pp. 230-265. (Correction *ibid*. 2-43, 544-546.)
7. P. Rosenbloom, A new framework for computer science and engineering. IEEE *Computer*: November 2004, pp. 31-36.
8. J. Wing, Computational thinking. ACM *Communications* **49**: March 2006, pp. 33-35.
9. J. Wing, Five deep questions in computing. ACM *Communications* **51**: January 2008, pp. 58-60.
10. P. Denning, D. Comer, D. Gries, M. Mulder, A. Tucker, J. Turner, and P. Young. Computing as a discipline, ACM *Communications* **32**(1), January 1989, pp. 9-23. A condensed version was published in IEEE *Computer*, February 1989.
11. ACM, *Curriculum 2001 Final Report*, Available: <acm.org/education/education/education/curric_vols/cc2001.pdf>
12. P. Denning, Is Computer Science Science? ACM *Communications* **48**: April 2005, pp. 27-31.
13. P. Denning, Computing is a natural science, ACM *Communications* **50**: July 2007, pp. 13-18.
14. P. Denning, Great Principles of Computing, ACM *Communications* **46**: November 2003, pp. 15-20.
15. D. Knuth, Computer Programming as an Art, ACM *Communications* **17**: December 1974, pp. 667-673.