The Computing Field: Structure

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

April 2008
(rev 9/14/08)

Abstract: This article examines the development of the computing field. Our account considers the computing field in four stages: infancy (1935-1950), childhood (1950-1970), adolescence (1970-1990), and young adulthood (1990-2010). The computing profession is a product of the fourth stage. The relationships between computing and other fields are vitally important.

Keywords: computing field, computing profession, IT profession, computer science, computer engineering, software engineering, information systems, computational science

The computing field has grown enormously since its inception in the 1930s. It began with the marriage of mathematical logic and digital electronics. It has matured into a complex of fields gathered under the large umbrella called computing (in the United States), informatics, and sometimes information technology (IT). As the field has grown, various individuals and groups have offered snapshots, which give their perceptions of its structure at their times. Considered in a sequence, these snapshots become a fascinating, animated story of how the field organized to accommodate its growth and its challenges. Some of the contributors to the sequence are ACM (1968), National Academy of Science (1968), Hamming (1969), Wegner (1970), Forsythe (1970), Amarel (1971), Arden (1976), Denning (1989), Hartmanis (1992), Tucker (1996), and ACM (2001, 2005). (See Refs. 1-11.)

This article examines the development of the computing field. Our account considers the computing field in four stages of development:

Infancy	1935-1950
Childhood	1950-1970
Adolescence	1970-1990
Young Adulthood	1990-2010

We will conclude with a discussion of the importance of the relationships between computing and other fields.

Although many consider the computing field to be mature, we have avoided that term because we do not want to give the impression that the next stages are decline and death. James Burke (12) points out how all major scientific revolutions took a century or more before their full impacts were felt. The computing revolution still has another half century before its maturity.

Our focus is on the structure of the field. Details about subareas will be found in separate articles.

The Infancy of Computing (1935-1950)

Machine-aided calculation of mathematical functions can be traced back many centuries. Algorithms devised by Pascal, Leibniz, and Gauss were used extensively to create tables of trigonometric, logarithmic, and exponential functions used by astronomers, navigators, and engineers. In the 1830s, Charles Babbage offered an escape from the tedium and errors of hand-calculated tables. He built a Difference Engine that calculated these tables automatically using difference equations. In a few hours, the Engine could produce without error entire tables that used to take years to develop. Spurred by the Engine's success, he undertook the design of an Analytic Engine that would calculate general mathematical functions, not just ones that satisfied difference equations. His design, never completed, lay dormant for nearly seventy years.

In the 1920s, Vannevar Bush of the Massachusetts Institute of Technology (MIT) built a large mechanical Differential Analyzer that solved partial differential equations of the kind frequently encountered in engineering. Unlike Babbage's discrete Difference Engine, Bush's Analyzer was analog.

In 1939, John Atanasoff built the first digital electronic computer at the Iowa State University. In the late 1930s, Konrad Zuse in Germany built calculating machines, which culminated with an all digital computer ("Z4") in 1941; but the German government did not take it seriously. Also in the 1930s, Alan Turing of the United Kingdom became interested in what such calculating machines could actually do. In his famous 1936 paper, he introduced an abstract computing device (now called a Turing machine), showed how to build a universal machine that could simulate any other, and showed that a centuries-old decision problem about the halting of computations could not be solved by any computing machine. Turing dashed the hopes of mathematicians that there might be some "by-inspection" method of determining whether a computation halts. He did this by demonstrating that the very process of "inspection" is inherently computational. Computation, said Turing, is fundamentally unavoidable (13).

Soon thereafter, Turing joined the team at Bletchley Park to design a computer to crack the German Enigma code. Because the British government classified that project for 25 years after the war, we did not know until around 1970 that Turing had helped build one of the first electronic computers.

At the start of World War II, the US and UK governments took a keen interest in electronic computing machines. The first of these machines were used to

calculate ballistic firing tables for the many new munitions that were being designed for the war. These governments commissioned electronic computer projects at many universities including Pennsylvania, Princeton Institute for Advanced Studies, MIT, Harvard, Cambridge, and Manchester. Those projects all had tremendous impact on the design of computing machines. Those universities also established courses of study in the new field.

John Mauchly and Presper Eckert, who were the builders of the Pennsylvania machine ENIAC, founded the Univac company to build computers for business. Univac delivered the first commercial electronic computer to the US Census Bureau for analysis of the 1950 census data. IBM soon expanded from its business machines market into the electronic computing market as well. These first machines were delivered with great fanfare. The newspapers called them "electronic brains". The nascent computing industry developed rapidly in the 1950s. Many universities offered courses in computing, mostly in electrical engineering or mathematics departments, and a few in business schools. These courses focused on the design of digital circuits, programming machines, processing data, and the theoretical limits of computation.

Many of the people involved in the first projects came together in 1947 to found the Association for Computing Machinery (ACM), which was the first professional society in the new field.

The digital electronic computer married three historical lines: mathematical logic, engineering, and science. The original teams included experts in all three fields. Mathematical logic brought notations for algorithms, universal machines, and mapping from logic formulas to physical switching circuits. Engineering brought passionate know-how for mechanical calculation and much expertise in electronics and electro-mechanical systems. Science brought a wealth of applications and methods for predicting the behavior of physical systems from their computational models. The imprint of these lines is still felt today. We will discuss them again later.

Computing's Childhood (1950-1970)

By 1950, the first computer building projects had succeeded and had stimulated a widening interest in the new technology. Over the next 20 years, the computing industry invented many technologies in programming languages, computer architecture, storage systems, time sharing, virtual memory, remote access, database, graphics, and robotics.

The academic world plunged in as well by creating courses in computers and computation, first as limited offerings, then as specializations within existing degree programs, and finally as separate departments with their own degree programs. The first computing degree was offered in the Moore School at University of Pennsylvania in the late 1950s. The first two computer science academic departments were founded at Purdue and Stanford in 1962. After that, academic departments in computer science (CS), computer engineering (CE), and information science (IS) sprung up like weeds in schools of science, engineering,

and business respectively. The number of CS and CE departments grew steadily, passing 160 in the early 1980s, 180 in the early 1990s, and 200 in the early 2000s.

In the mid 1960s, the ACM undertook the task to define curriculum recommendations for schools that wished to offer degrees in computer science. They wanted to establish a solid intellectual core for the new field and some minimum standards for all computer science degrees. Their report, titled ACM Curriculum 1968, said that the field consisted of three main parts:

Information structures and processes,
Information processing systems, and
Methodologies.

The methodologies included design approaches for software and applications. The core material was mostly the mathematical underpinnings for the parts listed above:

Algorithms
Programming
Data Structures
Discrete Math
Logic Circuits
Sequential Machines
Parsing
Numerical Methods

Many computer science departments adopted these recommendations (1).

During this time, the different orientations of the key players -- science, engineering, and business -- led to some interesting terminological confusion that was not resolved until the late 1980s. They argued over the definition of computer science and whether computer science was the appropriate title. Is the emphasis on the construction and analysis of algorithms (CS), the construction of fast and reliable machines (CE), or on information processes (IS)? Some leaders tried to find definitions that would encompass all three perspectives. The most famous was the definition by Alan Perlis with help from Gordon Newell and Herb Simon: "Computer science is the study of phenomena surrounding computers." (5)

The title "computer science" seems to have originated with the writings of John von Neumann, who recognized extensive ways computing could advance science, for example computational methods for hydrodynamics and Monte Carlo simulations of physical phenomena. He advocated for a science-based approach to the architecture of computers. Not surprisingly, the computer science title did not sit well with computer engineers or information systems people; and it also did not sit well with physical scientists, who felt that science is about natural phenomena, not man-made phenomena.

The Europeans avoided much of this wrangling by naming the field "Informatics"; this title can accommodate all three branches. They thought the field was about information processes; computers are tools for implementing and studying information processes. However, "informatics" never took hold in the US. The wrangling over title and scope lasted well into the 1980s.

The Adolescence of Computing (1970-1990)

This period was a time of great technological advances in computing. The computer chip was invented and became the mainstay of ever-advancing computing power. Intel's Gordon Moore observed that the number of transistors on a chip doubled every two years; this trend became known as Moore's Law. Computing power per chip increased by about 1 million over that time. The relentlessly advancing chip gave us the personal computer revolution and made computers ubiquitous.

The Internet was another major advance during that time. Its first nodes came online at the beginning of 1970. It grew very slowly at first, reaching about 200 nodes by 1980. Then it started to take off, reaching about 200,000 nodes by 1990. The Internet and the personal computer advanced together, accelerating each other's progress.

The major computer makers of the 1950s and 1960s mostly disappeared, except for IBM, and they were replaced by new generations that excelled with networked personal computers --- Apple, Microsoft, and Sun are conspicuous examples. All the major technologies advanced -- new programming languages, machine architectures, networks, operating systems, databases, robots, and graphics. Online services appeared, such as bulletin boards, software distribution servers, reprint distribution servers, X windows, authentication services, and more.

The disciplines CS, CE, and IS also flourished during this time. Many students were attracted to these fields so that they could make their careers out of helping advance the technologies.

In the mid-1970s the National Science Foundation (NSF) commissioned COSERS, which was the Computer Science and Engineering Research Study (7,14). Their objective was to take stock of all the research in the computer science and engineering (CS&E) fields and make its key ideas accessible to a wide audience. They defined CS&E as the field that studied algorithms and representations, always looking toward efficient implementations, and driven by the question, "What can be automated?" They reported ten subject areas for CS&E research:

Artificial intelligence
Data management
Hardware systems
Numerical computation
Operating systems
Programming languages
Software methodology
Theory of computation
Special topics
Applications

By the mid-1980s, the advancing computing technologies fostered a transformation in science. In 1982, Ken Wilson won the Nobel Prize in Physics by discovering a computational method that explained the way materials

changed their magnetic states. He became a strong advocate for using computation to advance science and coined the phrase "computational science". By the late 1980s, scientists from many fields had joined the refrain: Computation had become the third paradigm of science, joining the traditional theory and experiment. Many leading scientists articulated "grand challenge problems" in science, which they believed were very hard but would yield to advanced supercomputing power. These problems included the design of materials from first principles, artificial intelligence, full simulation of aircraft in flight, and design of new drugs. The US government created an interagency effort called High Performance Computing and Communication (HPCC) initiative to provide funds to build supercomputers and apply them to the grand challenge problems. Many computer scientists collaborated in these projects. The US Congress passed an HPCC act in 1991.

As the opportunities for computer science to engage with grand challenge problems enlarged, leaders of the computing field became painfully aware that their field had an external image "CS=programming" (computer science equals programming). They found that computer scientists and engineers were welcomed to the grand challenge projects mainly because of their expertise at programming, but not for their prowess in the scientific research.

The way this image developed is interesting. Ever since the early computer projects, computer scientists and engineers realized that a major part of their lives would be finding and correcting mistakes in their programs. The bigger the software system, the less reliable it was. In 1968, a group of software leaders came together at a NATO conference to consider how to address the growing "software crisis". They concluded that software could be made reliable only if software systems were put together with the same rigor as other engineering systems. They called for the creation of a new field, which they called software engineering. Software engineering has made some enormous advances in tools and methods, but it has always lagged behind the size and complexity of systems that we reach for. Thus, the software crisis has been an enduring crisis.

Curriculum 68 recognized that programming is a major activity of computer scientists and put it at the center of their curriculum recommendations. Just a few years later, Donald Knuth and Edsger Dijkstra both proclaimed they were programmers and made algorithms design and analysis into a high art. They made it intellectually respectable to be programmers.

Unfortunately, the trend in the outside world was heading in the reverse direction. Programming was seen as tedious work. Mildly derogatory words like "code jock" and "hacker" became synonyms for "programmer". The US Bureau of Labor Statistics defined programmers narrowly, essentially as coders. By the late 1980s, insiders saw programming as a noble calling and outsiders saw it as low-level drudgework. This conflict threatened future jobs of computer science and engineering graduates as well as scientific collaborations.

In 1987, ACM and Institute for Electrical and Electronics Engineers (IEEE) joined together to create a task force to defeat the narrow notion that "CS=programming". In early 1989, they published an influential report, "Computing as a discipline" that made three major contributions (8):

1. It recognized that computer science and computer engineering had a common core of knowledge and used the term "computing" to encompass both. Thus "computing discipline" was shorthand for "the discipline of computer science and engineering." The term computing has since become widely accepted and is on par with the European Informatics.
2. It recognized that the three paradigms of mathematics, science, and engineering played major roles in the field. Mathematics brings the rigor of clear notation and the power of logic and deduction. Science brings experimental methods, modeling, validation of hypotheses, and induction. Design brings order and reliability to the processes of constructing large systems. The unique flavor of computing comes from the constant interplay among these ways of thinking.
3. It recognized nine core areas of computing. The nine topic areas all had their own identities, which included technical expertise, literature, and professional organizations.

Table 1 depicts the 9x3 matrix model of the computing field offered by the report. Theory, abstraction, and design were used in the report for the mathematics, science, and engineering paradigms respectively. The report gave details about what ideas and technologies fit into each of the 27 boxes in the matrix. It became the basis for a major ACM/IEEE curriculum revision in 1991.

Although this effort had a strong internal influence on the curriculum, it had little external influence on the perception that "CS=programming". In fact, that perception was alive and well in 2004 and was causing considerable difficulty in recruiting majors (15).

Table 1: Matrix Model of Computing Discipline, 1989.

Topic Area	Theory	Abstraction	Design
1 Algorithms & Data Structures			
2 Programming Languages			
3 Architecture			
4 Operating Systems and Networks			
5 Software Engineering			
6 Databases & Information Retrieval			
7 Artificial Intelligence & Robotics			
8 Graphics			
9 Human Computer Interaction			

Computing's Young Adulthood (1990-2010)

The World Wide Web was launched in 1989. It became visible widely in the computing research community by 1991 and among all Internet users by 1994 with the first portable and free browser (Mosaic). It completely transformed the Internet and the way we thought about computing. It expanded our perceptions of the size of an information system. It enabled electronic commerce and Web-based businesses and services. It brought the problem of search into prominence. It created controversies over free distribution and copyrights. It enabled computing on massive scales (grids). It enabled massive multiplayer games. It stimulated new courses of study at universities.

In 1998, the ACM launched an "IT profession" initiative, in which it recognized that the field had evolved from a discipline to a profession (16). The initiative responded to the growing interest in the industry for professional standards (especially in safety critical systems), organized professional bodies representing various specialties, and the university movement to establish degree programs in information technology. The ACM concluded that the computing met the following four criteria for a profession:

1. A durable domain of human concerns.
2. A codified body of principles (conceptual knowledge).
3. A codified body of practices (embodied knowledge including competence).
4. Standards for competence, ethics, and practice.

The ACM concluded that these criteria were met for the computing field and that it was time for the ACM to configure itself to support the field as a profession.

The ACM made an inventory of the organized groups that participated in the field (Table 2) (17). IT professionals are a much larger and more diverse group than computer scientists and engineers. They have organized affinity groups in at least 42 specialties of three categories. The first category comprises the major technical areas of IT and spans the intellectual core of the field. The second category comprises other well-established fields that are intensive users of IT; they draw heavily on IT and often make novel contributions to computing. The third category comprises areas of skill and practice necessary to keep support the IT infrastructures that everyone uses.

Several important conclusions can be drawn as follows: (1) The IT profession has broad scope, which includes subfields from science, engineering, and business; (2) the players share a common base of science and technology but have distinctive professional practices; (3) many players are willing to identify with the IT field but not with the computer science discipline; and (4) strong leadership from the professional societies is needed to keep these players united under the common IT identity. The ability of the IT field to resolve broad, systemic problems such as software quality, basic research, and professional lifelong education requires extensive cooperation among the players. Several universities established IT departments and schools to address the needs of the profession directly (18,19).

Table 2: The Profession of Information Technology

IT-Core Disciplines	IT-Intensive Disciplines	IT-Supportive Occupations
Artificial intelligence Computer science Computer engineering Computational science Database engineering Graphics Human computer interaction Network engineering Operating systems Performance engineering Robotics Scientific computing Software architecture Software engineering System security	Aerospace engineering Bioinformatics Cognitive science Cryptography Digital library science E-commerce Economics Genetic engineering Information science Information systems Public Policy and Privacy Quantum Computing Instructional design Knowledge engineering Management information systems Materials Science Multimedia design Telecommunications	Computer technician Help desk technician Network engineer Professional IT trainer Security specialist System administrator Web services designer Web identity designer Database administrator

The talk about "profession" led to a new round of terminological confusion. The IT profession is a social structure that includes many disciplines; but it is not a discipline in its own right. IT is not a field of research; the core disciplines (left column) and partner disciplines (middle column) attend to the research. To what does the term "computing field" refer in this context?

This was one question on the minds of ACM and IEEE when in 1999 they undertook a major review of curriculum recommendations. In their report, Computing Curriculum 2001 (CC2001), they focused on the core specialties (Left column of Table 2) (11). They identified the "computing discipline" ("informatics" outside the US) with these six academic specialties:

EE -- Electrical Engineering
CE -- Computer Engineering
CS -- Computer Science
SWE -- Software Engineering
IS -- Information Systems
IT -- Information Technology

Here the term "IT" can be confusing. It refers to a set of degree programs that focus on organizational applications of computing technology; it does not refer to the entire profession. The ACM/IEEE used the map of Fig. 1 to illustrate how the field had changed and to suggest the choices available to students interested in hardware, software, or organizational issues.

They also constructed a "body of knowledge" representing the common core of the six disciplines in the family. The body of knowledge listed 14 core areas of the computing field (Table 3), up from the 9 areas listed by ACM just a dozen years before. Under the 14 major headings, they listed 131 subareas, of which 63

were designated as core. This reorganization was a major expansion over previous summaries of the field. Theory, Abstraction, and Design were still present, but less prominent (11).

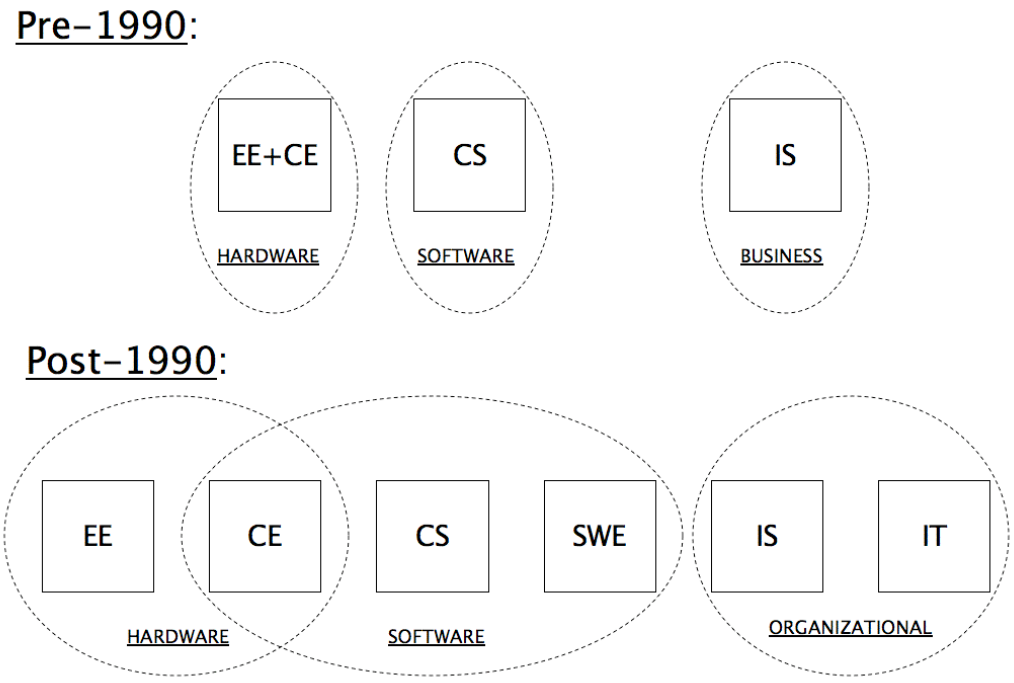

Figure 1. Student's view of academic offerings in computing (ACM 2005).

Thus we recognize in the maps of the field from around 2001 a two-dimensional structure: One dimension maps a social structure of professional subfields, the other maps a technical structure represented by the body of knowledge.

All this left the term "IT" referring to both the profession and to a degree program, and the term "computing field" with no precise definition. Perhaps that is just as well: Fuzzy terms are more easily reinterpreted as the field matures.

The CC2001 report was still fresh when another challenge appeared. The computing field faced another paradigm shift, just as it had done in the mid-1980s with computational science. Now many fields of science claimed information processes in their deepest structures. Because some information processes occur in nature, computing was recognized as a natural science. That claim killed the Perlis notion that computing is about phenomena that surround computers. It forced a resurrection of the older notion that computing is the work of information processes. The computer is a tool, not an object of study. In reality, it had always been this way: Computing always had natural and artificial flavors (20, 21).

Table 3: Model of Computing Discipline, 2001.

Topic Area	T	A	D
1 Discrete Structures			
2 Programming Fundamentals			
3 Algorithms and Complexity			
4 Architecture and Organization			
5 Operating Systems			
6 Net-Centric Computing			
7 Programming Languages			
8 Human Computer Interaction			
9 Graphics and Visual Computing			
10 Intelligent Systems			
11 Information Management			
12 Social and Professional Issues			
13 Software Engineering			
14 Computational science			

Even the two-dimension structures of the curriculum 2001 were not flexible enough to deal with this shift. A movement to understand computing in terms of great scientific principles was launched and gained momentum (22). The great principles framework complements but does not replace the other ways of looking at the field. It organizes the principles of computing into the seven categories:

Computation
Communication
Coordination
Recollection
Automation
Evaluation
Design

See the article in this encyclopedia on Great Principles of Computing for details on this framework.

It is an irony that computer science, the discipline that gave birth to the IT profession and computing field, is not the driving force. The field is being driven by the large numbers of user pragmatists, which include many powerful business, civic, government, and industry leaders. Computer scientists no longer "control" the field. Their main role is to advance the scientific, engineering, and mathematical knowledge of computing.

Relations with Other Fields

A hallmark of the computing field has been its close relations with numerous other fields. The reason is not hard to understand: Information processes are part of most fields. All the taxonomies of the field from the 1960s to the present day contain some sort of entry for "applications", which expressly acknowledges these many links.

Computer science has always had close bonds with mathematics. Mathematical logic, the theorems of Turing and Gödel, Boolean algebra for circuit design, and algorithms for solving equations and other classes of problems in mathematics played strong roles in the early development of the field. Conversely, computer science has strongly influenced mathematics; for example, proofs of existence are often formulated as algorithms that construct or select a mathematical object. In some cases, computers have been essential to mathematics; for example, the solution of the four-color theorem relied on a program that searched a large finite number of cases for counterexamples. Within computer science the powerful and far-reaching theory of complexity is a mathematical tour-de-force, and "Is P=NP?" is one of the hard questions of mathematics. For these reasons, some observers like to say that computing is a mathematical science.

The bond between engineering and computer science has been much stronger than between many natural science disciplines and their engineering counterparts -- for example, chemical engineering and chemistry, aircraft design and fluid dynamics, pharmacy and biology, and materials engineering and physics. This bond exists because computer science has a strong heritage in electrical engineering and because many algorithmic methods were designed originally to solve engineering problems. Examples include electronic circuits, telecommunications, engineering graphics, engineering design, systems engineering, fabrication, and manufacturing. Conversely, computers have become indispensable in many engineering disciplines -- for example, circuit simulators, finite-element simulators, flow-field simulators, graphics, computer-assisted design (CAD) and computer-assisted manufacturing (CAM) systems, computer-controlled tools, and flexible manufacturing systems. For these reasons, some observers like to say that computing is an engineering field.

There has always been a bond between the physical sciences and computer science. Computers were always envisioned as tools for scientific and engineering calculations. In the late 1980s, leaders of physics, chemistry, biology, geology, seismology, astronomy, oceanography, and meteorology brought to prominence certain very hard, "grand challenge" problems that demand massive high-speed computations, which are performed on new generations of massively parallel computers with massively parallel algorithms. These problems include crystalline structure, quantum electrodynamics, calculation of chemical properties of materials from the Schrödinger equation, simulation of aircraft in flight, exploration of space, global climate modeling, oil exploration, models of the universe (cosmology), long range weather forecasting, earthquake prediction, turbulent fluid flow, and human genome sequencing. Those leaders proclaimed that computation had become a third paradigm of science, joining theory and experimentation. After 2000, this bond deepened as those fields recognized

information processes in their deep structures and entered into many collaborative relationships with computing people (20). For these reasons, some observers like to say computing is a science.

Who's right? They all are. The computing field is rich and deep, with a strong heritage in mathematics, engineering, and sciences. In addition to the influences of these heritages, interactions with other fields are woven into the discipline itself. Here a few examples:

- The computer science problem "Is P=NP?" has been listed as one of the most difficult unsolved problems in modern mathematics. The computer science collaboration in discrete mathematics has placed modern discrete mathematics on a par with the older continuous mathematics.
- Computer science has contributed advanced methods in computing over adaptive grids that have enabled advances in the design of buildings, aircraft, automobiles, and mechanical parts.
- Computer science has worked closely with computer engineering on architectures for parallel computation, neural computation, functional computation, and dataflow computation.
- Library science is concerned with archiving texts and organizing storage and retrieval systems to give efficient access to texts. Digital library systems have changed libraries from book repositories to electronic data centers, which are accessible throughout the world via the Internet. Libraries are concerned with advanced search technologies to locate information in the Internet. They have a special concern with data migration from older storage media onto newer ones.
- Medicine uses computer models and algorithms in ingenious ways to diagnose and treat diseases. Modern imaging methods such as magnetic resonance scans, coronary scans, and tomography have drawn heavily on computer science. Medical researchers use computer models to assist them in tracking mutations of viruses and in narrowing the search for new molecules that may be effective drugs. The Human Genome Project used large distributed databases and new kinds of string-matching algorithms to aggregate the tens of thousands of DNA-sequencing experiments.
- Biology is deeply concerned with the meaning of DNA and the mechanics of DNA transcription. In collaboration with computer scientists, biologists have developed new algorithms for classifying and searching genome databases (which are enormous). They have been studying DNA transcription as a form of computation. They have demonstrated how problem statements can be encoded into DNA molecules and solutions generated in their chemical interactions.
- Physicists regard quantum functions as information waves whose interactions generate physical particles and forces. These ideas have led to quantum computation, a new form of computation in which information is represented with quantum waves, and to quantum cryptography.

- Materials chemists regard molecules as manifestations of forces described by the Schrödinger equation. They can design new materials by computing and testing molecular structures.
- Management science uses computer models to plan and forecast economic conditions for business. They store business records in databases from which they manage complex customer relations and enact complex commitments.
- Economics views markets as information flow and exchange media. They use computer models to forecast economic conditions and to evaluate the possible effects of macroeconomic policies.
- Forensics uses computer models and large databases to identify evidence and make inferences about the criminal intents of users.
- Psychology, xognitive, and behavioral sciences are concerned with understanding human thought and emotions. They use computer models to gain insight into the operation of human brains and nervous systems and to design effective interventions for human problems.
- Linguistics is concerned with using computers to recognize speech, translate between languages, and to understand the role of language in human affairs.
- Social scientists use graph and clustering algorithms for social network mapping, which aids in understanding trust, influence, power, and information flow in social systems.
- Philosophy studies the way people acquire knowledge, create social realities, and act morally and ethically. Philosophers have contributed much to the debates on whether machines can think or whether formal models are sufficient for dependable software systems. The subdiscipline of language action has contributed much to our understanding of how people carry out work in organizations and has helped give birth to the workflow industry. The technologies of "virtual realities" have rekindled debates on the nature of reality and the worlds in which people live.
- Humanities uses computers extensively to correlate and search through historical artifacts that can be represented digitally. One of the more colorful examples is the use of computers to determine authorship of historical texts, such as Shakespeare's plays.

This list is hardly exhaustive. The number of contacts between computing and other disciplines grows each year. The people who bridge between computer science and other fields are doing some of the most innovative work. These interactions are likely to be the hallmark of computing in the future (23-25).

Paul Rosenbloom has mapped the structure of the relationships between computing -- meaning computer science and engineering -- and other fields (26). It is very rich. The non-CS&E fields are of three kinds:

- Physical sciences (P), which focus on nonliving matter,
- Life sciences (L), which focus on living matter, and
- Social sciences (S), which focus on humans and their societies.

The relations between computing and these other fields are of three kinds:

- Implementation (/), in which technology from one field is used to implement a function in another field.
- Interaction (•), in which two fields collaborate as peers.
- Embedding [ ], in which a fragment of one field is integrated into another.

He constructed a chart (Table 4) that provides examples of fruitful relationships in all the categories above.

Table 4: Interactions between computing and other fields

	C+P	C+L	C+S	C+C
Implementation (/): Technology from the physical sciences (P), life science (L), or social sciences (S) is used to implement computation (C) or computation is used to implement (possibly a model or function of) an aspect of one of the domains.				
C/*	C/P: Silicon and quantum computing	C/L: Biological and neural computing	C/S: Wizard of Oz	C/C: Languages, compilers, operating systems, emulation
***/C**	P/C: Modeling and simulation, data/information bases	L/C: Artificial life, bioinformatics, systems biology	S/C: Artificial intelligence	
Interaction (•): A symmetric relationship in which two domains interact as peers.				
C•* ***•C**	C•P and P•C: Sensors, effectors, robots, peripherals	C•L and L•C: Biosensors	C•S and S•C: Human-computer interaction, authorization	C•C: Networking, security, parallel computing, grids
Embedding ([ ]): Some fragment of one domain is embedded in another.				
C[*]	C[P]: Analog computing	C[L]: Autonomic systems	C[S]: Immersion	C[C]: Embedded monitoring and testing
***[C]**	P[C]: Embedded computing	L[C]: Cyborgs	S[C]: Cognitive prostheses	

Conclusion

We have traced the evolution of the structure of the computing field from its inception to the early twenty-first century. In the beginning (1930s), it was an amalgam of mathematics, engineering, and science. By the 1960s, it had an emerging identity as "computer science and engineering" and a simple internal structure of core technologies. By the 1980s, it had responded to the explosive growth of computing by enlarging the basic structure and had begun to deal with the challenge of computational science. By the 1990s, it had to contend with the Internet and the Web, which brought ordinary users and businesses into computing, and fostered growth of three other computing-related fields -- software engineering, information systems, and information technology. By the 2000s, recognizing itself as a profession, it focused on managing the core identify and the interactions between the computing-related disciplines and all the other users of computing. It began to meet the new challenge of its acceptance as a natural science by developing a great principles framework of its scientific fundamentals.

Our review of the history of computer science reveals an interesting progression of definitions for computer science:

Study of information processing (1940s)
Study of phenomena surrounding computers (1960s)
Study of what can be automated (1970s)
Study of computation (1980s)
Study of information processes, both natural and artificial (2000s)

Over time, the definition of computer science has been a moving target. It has cycled back to a definition that resembles our roots because of recent affirmations from the physical sciences that they study information processes found in nature.

Paradoxically, just as it was entering its adulthood in the early 2000s, the computing field experienced a sudden decline in student enrollments -- 50% drop from 2002 to 2007 -- at a time when industry demand for graduates was growing, and prospects for novel collaborations were higher than ever. The decline seemed related to an external identity that most workers in the field were programmers, system administrators, and network configuration engineers. The new configurations and relationships outlined here set the stage for a new period of innovation and growth of computing.

Bibliography

1. ACM, Curriculum 68, William F. Atchison, et al, eds., ACM *Communications* **11**(3), 151-197, 1968.
2. National Academy of Sciences, *The mathematical sciences: A Report.* Publication 1681, Washington, D. C., 1968.
3. R. W. Hamming, One man's view of computer science, *ACM Turing Lecture. J. ACM* **16**(1), 1-5, 1969.

4. P. Wegner, Three computer cultures -- computer technology, computer mathematics, and computer science, in *Advances in Computers 10*, W. Freiberger (ed.). New York: Academic Press, 1970.
5. G. Forsythe, B. A. Galler, J. Hartmanis, A. J. Perlis, and J. F. Traub, Computer science and mathematics, ACM *SIGCSE bulletin* 2(4), 1970. Available: http://doi.acm.org/10.1145/873661.873662
6. S. Amarel, Computer science: a conceptual framework for curriculum planning, *ACM Communications* 14(6), 1971.
7. B. Arden, The computer science and engineering research study. ACM *Communication*s **19**(12), 670-673, 1971.
8. P. Denning, D. E. Comer, D. Gries, M. C. Mulder, A. Tucker, A. J. Turner, P. R. Young, Computing as a discipline, ACM *Communications* **32**(1), 9-23, 1989.
9. J. Hartmanis, et al. *Computing The Future.* Washington, D.C.: National Academy of Science Press, 1992.
10. A. Tucker Jr. and P. Wegner, Computer Science and Engineering: The discipline and its impact, in *Handbook of Computer Science and Engineering*. Boca Raton, FL: CRC Press, Chapter 1, 1996.
11. ACM-IEEE, Curriculum Recommendations, 2001 and 2005. Available: www.acm.org/education/curricula-recommendations
12. J. Burke, *The Day the Universe Changed*. Boston, MA: Back Bay Books, 1995.
13. A. Turing, On computable numbers with an application to the Entscheidungsproblem, *Proc. of the London Mathematical Society*, *Series 2*, 42, 1936, pp. 230-265.
14. B. Arden (ed.), *What can be automated? -- The Computer Science and Engineering Research Study*. Cambridge, MA: The MIT Press, 1980.
15. P. Denning, The field of programmers myth, ACM *Communications* **47**(7), 15-20, 2004.
16. P. Denning, Computing the profession, *Educom Review* **33**(6), 26-30 and 46-59.
17. P. Denning, Who are we? ACM *Communications* **44**(2), 15-19, 2001.
18. P. Denning, The IT schools movement, ACM *Communications* **44**(8), 19-22, 2001.
19. N. Holmes, Fashioning a Foundation for the Computing Profession, IEEE *Computer* (July), 97-98, 2000.
20. P. Denning, Is computer science science? ACM *Communications* **48**(4), 27-31, 2005.
21. P. Denning, Computing is a natural science, ACM *Communications* 50(7), 13-18, 2007.
22. P. Denning, Great Principles of Computing, ACM *Communications* **46**(11), 15-20. 2003.

23. P. Denning and R. Metcalfe (eds.) *Beyond Calculation: The Next 50 Years of Computing*, New York: Springer-Verlag, 1997.

24. P. Denning, *Talking Back to the Machine: Computers and Human Aspiration*, New York: Springer-Verlag, 1999.

25. P. Denning, *The Invisible Future.* New York: McGraw-Hill, 2001.

26. P. Rosenbloom, A new framework for computer science and engineering, IEEE *Computer* (November), 31-36, 2004.

