Handbook of Applications of Chaos Theory

by

Christos H Skiadas and Charilaos Skiadas, Editors

Foreward

We are delighted to introduce the Handbook of Applications of Chaos Theory.....

Preface

The last decades advances on Chaos Theory resulted in numerous applications in various scientific and applied fields.....

Contents

Part I: This is a Part

Symbol Description

α	To solve the generator maintenance scheduling, in the past, several mathematical techniques have been applied.
σ^2	These include integer programming, integer linear programming, dynamic programming, branch and bound etc.
$\sum$	Several heuristic search algorithms have also been developed. In recent years expert systems,
abc	fuzzy approaches, simulated annealing and genetic algorithms have also been tested.
$\theta\sqrt{abc}$	This paper presents a survey of the literature
ζ	over the past fifteen years in the generator
∂	maintenance scheduling. The objective is to
sdf	present a clear picture of the available recent literature
ewq	of the problem, the constraints and the other aspects of
bvcn	the generator maintenance schedule.

I

This is a Part

1

Computer Science and Engineering: The Discipline and Its Impact

Allen B. Tucker, Jr.
Bowdoin College

Peter Wegner
Brown University

1.1 Introduction

The discipline and profession of computer science and [**?**] engineering (CS&E) has undergone dramatic changes in its short 50-year life. As the field has matured, new areas of research and development have emerged and joined with older areas to revitalize the discipline. In the 1930s, fundamental mathematical concepts of computing were developed by Turing and Church. Early computers implemented by von Neumann, Eckert, Atanasoff, and others in the 1940s led to the birth of commercial computing in the 1950s and to numerical programming languages like Fortran, commercial languages like COBOL, and artificial-intelligence languages like LISP. In the 1960s the rapid development and consolidation of the subjects of [**?**] algorithms, data structures, databases, and operating systems formed the core of what we now call traditional computer science; the 1970s saw the emergence of software engineering, structured programming, and object-oriented programming. The emergence of personal computing and networks in the 1980s set the stage for dramatic advances in computer graphics, software technology, and parallelism. This Handbook aims to characterize computing in the 1990s, incorporating the explosive growth of networks like the World Wide Web and the increasing importance of areas like human–computer interaction, computational science, and other subfields that would not have appeared in such an

This introductory chapter reviews the evolution[**?**] of CS&E during the last two decades. It introduces those fundamental contemporary themes that form the nucleus of the subject

matter and methodologies of the discipline, identifying the social context and scientific challenges that will continue to stimulate rapid growth and evolution of CS&E into the next century. Finally, it provides an overview of the discipline of CS&E, serving as a conceptual introduction to the ten major sections and 122 chapters and appendices that constitute the entire Handbook. These ten sections, corresponding to ten major subject areas, reflect a useful classification of the [**?**] subject matter in CS&E:

- Algorithms and Data Structure
- Architecture
- Artificial Intelligence and Robotics
- Computational Science
- Database and Information Retrieval
- Graphics
- Human–Computer Interaction
- Operating Systems and Networks
- Programming Languages
- Software Engineering

Section 1.2 of this chapter presents a brief history of the computing industry and the parallel development of the computing curriculum. Section 1.3 frames the practice of CS&E in terms of four major conceptual paradigms: theory, abstraction, design, and the social context. Section 1.4 [**?**] identifies the "grand challenges" that promise to extend the field's vitality and reshape its definition for the next generation and beyond, and section 1.5 summarizes the contents of the ten sections of the Handbook.

definition for the next generation and beyond, and section 1.5 summarizes the contents of the ten sections of the Handbook.

definition for the next generation and beyond, and section 1.5 summarizes the contents of the ten sections of the Handbook.

definition for the next generation [**?**] and beyond, and section 1.5 summarizes the contents of the ten sections of the Handbook.

This Handbook is designed as a professional reference for researchers and practitioners in the field. Readers interested in exploring specific subject topics may prefer to move directly to the appropriate section of the Handbook. To facilitate rapid inquiry, the Handbook contains a Table of Contents and three indexes (Subject, Who's Who, and Key Algorithms and Formulas) for immediate access to specific topics at various levels of detail.

1.2 Growth of the Industry and the Profession

The computer industry has experienced tremendous growth and change over the last several decades, and most recently some retrenchment. The transition that began in the 1980s, from centralized mainframes to a decentralized networked microcomputer–server technology, was accompanied by the rise and fall of major corporations. The old monopolistic, vertically integrated industry epitomized by IBM's comprehensive client services gave way to a highly competitive industry in which the major players changed almost overnight. In 1992 alone, emergent companies like Dell and Microsoft had spectacular profit gains of 77% and 53%. In contrast, traditional companies like IBM and DEC suffered combined record losses of $7.1 billion in the same year [Economist 1993]. The exponential decrease in computer cost and increase in power by a factor of two every eighteen months, known as Moore's law, shows no signs of abating, though underlying physical limits must eventually be reached.

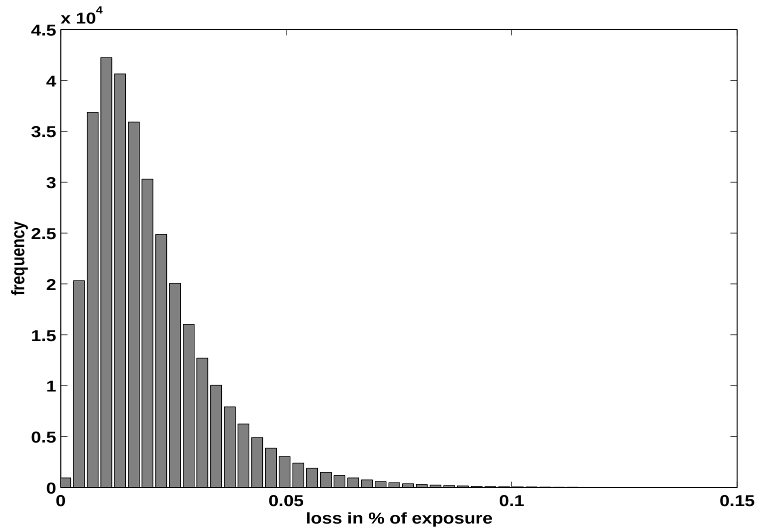

FIGURE 1.1 The doubling program in the GOTO language. The doubling program in the GOTO language. The doubling program in the GOTO language.

Example 1.1

This inside the Example Environment. This inside the Example Environment.

This is normal text. This is normal text. This is normal text. This is normal text. This is normal text. This is normal text. This is normal text. This is normal text. This is normal text. This is normal text. This is normal text. This is normal text.

FACT 1.1 *A component part for an electronic item is [?] manufactured at one of three different factories, and then delivered to the main assembly line. Of the total number supplied, factory A supplies 50%, factory B 30%, and factory C 20%. Of the components manufactured at factory A, 1% are faulty and the corresponding proportions for factories B and C are 4% and 2% respectively. A component is picked at random from the assembly line. What is the*

```
 1  PRINT 0
 2  GO LEFT
 3  GO TO STEP 2 IF 1 IS SCANNED
 4  PRINT 1
 5  GO RIGHT
 6  GO TO STEP 5 IF 1 IS SCANNED
 7  PRINT 1
 8  GO RIGHT
 9  GO TO STEP 1 IF 1 IS SCANNED
10  STOP
```

FIGURE 1.2 The doubling program in the GOTO language. The doubling program in the GOTO language. The doubling program in the GOTO language.

probability that it is faulty?

Overall, the rapid 18% annual growth rate that the computer industry had enjoyed in earlier decades gave way in the early 1990s to a 6% growth rate, caused in part by a saturation of the personal computer market. Another reason for this slowing of growth is that the performance of computers (speed, storage capacity) has improved at a rate of 30% per year in relation to their cost. Today, it is not unusual for a desktop computer to run at hundreds of times the speed and capacity of a typical mainframe computer of the 1980s, and at a fraction of the cost. However, it is not clear whether this slowdown in growth represents a temporary plateau or whether a new round of fundamental technical innovations in areas such as networking and human–computer interaction might again propel the computer industry to more spectacular rates of growth.*

Overall, the rapid 18% annual growth rate that the computer industry had enjoyed in earlier decades gave way in the early 1990s to a 6% growth rate, caused in part by a saturation of the personal computer market. Another reason for this slowing of growth is that the performance of computers (speed, storage capacity) has improved at a rate of 30% per year in relation to their cost. Today, it is not unusual for a desktop computer to run at hundreds of times the speed and capacity of a typical mainframe computer of the 1980s, and at a fraction of the cost. However, it is not clear whether this slowdown in growth represents a temporary plateau or whether a new round of fundamental technical innovations in areas such as networking and human–computer interaction might again propel the computer industry to more spectacular rates of growth.

Example 1.2

This inside the Example Environment. This inside

*Here goes another one

the Example Environment. This inside the Example Environment. This inside the Example Environment.

Solution 1.1

This goes inside the solution. This goes inside the solution. This goes inside the solution. This goes inside the solution. This goes inside the solution. This goes inside the solution. This goes inside the solution. This goes inside the solution.

Overall, the rapid 18% annual growth rate that the computer industry had enjoyed in earlier decades gave way in the early 1990s to a 6% growth rate, caused in part by a saturation of the personal computer market. Another reason for this slowing of growth is that the performance of computers (speed, storage capacity) has improved at a rate of 30% per year in relation to their cost. Today, it is not unusual for a desktop computer

THEOREM 1.1 *In most literature on PPDP, they [**?**] consider a more relaxed, yet more practical, notion of privacy protection by assuming limited attacker's background knowledge. Below, the term "victim" refers to the record owner being linked. We can broadly classify linking models to two families.*

$$var\widehat{\Delta} = \sum_{j=1}^{t} \sum_{k=j+1}^{t} var(\hat{\alpha}_j - \hat{\alpha}_k) = \sum_{j=1}^{t} \sum_{k=j+1}^{t} \sigma^2(1/n_j + 1/n_k). \tag{1.1}$$

One family considers a privacy threat occurs when an attacker is able to link a record owner to a record in a published data table, to a sensitive attribute in a published data table, or to the published data table itself. We call them record linkage*,* attribute linkage*, and* table linkage*, respectively. In all types of linkages, we assume that the attacker knows the QID of the victim. In record and attribute linkages, we further assume that the attacker knows the presence of the victim's record in the released table, and seeks to identify the victim's record and/or sensitive information from the table [**?**]. In table linkage, the attack seeks to determine the present or absent of the victim's record in the released table. A data table is considered to privacy preserved if the table can effectively prevent the attacker from successfully performing these types of linkages on the table [**?**]. Sections* **??**-**??** *study this family of privacy models.*

Solution 1.2

This goes inside the solution. This goes inside the solution. This goes inside the solution. This goes inside the solution. This goes inside the solution. This goes inside the solution. This goes inside the solution. This goes inside the solution.

To run at hundreds of times the speed and capacity of a typical mainframe computer of the 1980s, and at a fraction of the cost. However, it is not clear whether this slowdown in growth represents a temporary plateau or whether a new round of fundamental technical innovations in areas such as networking and human–computer interaction might again propel the computer industry to more spectacular rates of growth. or whether a new round of fundamental technical innovations in areas such as networking and human–computer interaction might again propel the computer industry to more spectacular rates of growth. or whether a new round of fundamental technical innovations in areas such as networking and human–computer interaction might again propel the computer industry to more spectacular

TABLE 1.1 Examples for illustrating attacks

Job	Sex	Age	Disease
Engineer	Male	35	Hepatitis
Engineer	Male	38	Hepatitis
Lawyer	Male	38	HIV
Writer	Female	30	Flu
Writer	Female	30	HIV
Dancer	Female	30	HIV
Dancer	Female	30	HIV

rates of growth. or whether a new round of fundamental technical innovations in areas such as networking and human–computer interaction might again propel the computer industry to more spectacular rates of growth.

1.2.1 Level 2/ B Head Curriculum Development

The computer industry's evolution has been strongly affected by the evolution of both theory and practice in the last several years. Changes in theory and practice are intertwined with the parallel evolution of the field's undergraduate and graduate curricula during the last three decades, and those curricula have, in turn, defined the conceptual and methodological framework for understanding the discipline itself.

Level 3/C Head

The computer industry's evolution has been strongly affected by the evolution of both theory and practice in the last several years. Changes in theory and practice are intertwined with the parallel evolution of the field's undergraduate and graduate curricula during the last three decades, and those curricula have, in turn, defined the conceptual and methodological framework for understanding the discipline itself.*

Curriculum Development (Lavel 4 Head)

The first coherent and widely cited curriculum for CS&E was developed in 1968 by the ACM Curriculum Committee on Computer Science [ACM 1968] in response to widespread demand for systematic undergraduate and graduate programs [Rosser 1966]. "Curriculum 68" defined computer science as comprising three main areas: information structures and processes, information processing systems, and methodologies. The first area included programming languages, data structures, and formal models of computation; the second computer architecture, compilers, and operating systems; the third numerical mathematics, file management, text processing, graphics, simulation, information retrieval, artificial intelligence, process control, and instructional systems. Curriculum 68 used this taxonomy to define computer science as a discipline and to provide concrete recommendations and guidance to colleges and universities in

$$\alpha + \beta = \gamma + a + \alpha \tag{1.2}$$

Lavel 5 Head Should be coded with a star. developing undergraduate, master's, and Ph.D. programs to respond to the widespread demand for computer scientists in research, education, and industry. Curriculum 68 stood as a robust and exemplary model for degree programs at all levels for a decade or more.

In 1978, a new ACM Curriculum Committee on Computer Science developed a revised

*This is the second footnote

and updated undergraduate curriculum [ACM 1978]. The "Curriculum 78" report responded to the rapid evolution of the discipline and practice of computing and to a demand for a more detailed elaboration of the computer science (as distinguished from the mathematical) elements of the courses that would comprise the core curriculum. Around the same time, the IEEE Computer Society developed a model curriculum for engineering-oriented undergraduate programs in CS&E [IEEE-CS 1976]. Updated and published in 1983 by the Computer Society as a "Model Program in Computer Science and Engineering" [IEEE-CS 1983], this curriculum was designed not only to define a course of study for computer science programs in engineering schools but also to meet a more extensive set of engineering accreditation criteria.

In 1988, the ACM Task Force on the Core of Computer Science and the IEEE Computer Society [ACM 1988] cooperated in developing a fundamental redefinition of the discipline of CS&E. Called "Computing as a Discipline," this report aimed to provide a contemporary foundation for undergraduate curriculum design by responding to the changes in computing research, development, and industrial applications in the previous decade. This report also acknowledged some fundamental methodological changes in the field. No longer could the "computer science = programming" model hope to encompass the richness of the field. Instead, three perspectives—*theory, abstraction,* and *design*—were used to characterize how various kinds of computer professionals and researchers did their work. These three points of view, those of the theoretical mathematician or scientist (theory), the experimental or applied scientist (abstraction, or modeling), and the engineer (design), were essential components of research and development throughout all the nine major subject areas (similar to the ten Handbook areas) into which the field was divided.

$$\alpha + \beta = \gamma \tag{1.3}$$

$$\alpha + \beta = \gamma \tag{1.4}$$

"Computing as a Discipline" led directly to the formation of a joint ACM/IEEE-CS Curriculum Task Force, which developed a comprehensive model for undergraduate curriculum design in the 1990s called "Curricula 91" [ACM/IEEE 1991]. Acknowledging that undergraduate computer science programs could be effectively supported in colleges of engineering, arts and sciences, and liberal arts, Curricula 91 proposed a core curriculum of common knowledge that undergraduate majors in any of these programs should cover. This core curriculum also contained sufficient theory, abstraction, and design content that students would become familiar with the fundamentally different but complementary ways of "doing" CS&E. It also ensured that students would gain a broad exposure to the nine major subject areas, and their social and ethical context. A significant laboratory component ensured that undergraduates gained significant abstraction (experimentation) and design experience.

TABLE 1.2 This is an Example of Table Title This is an Example of Table Title This is an Example of Table Title This is an Example of Table Title

Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Ball	19,221	4,598	9,188	3,209	3,200
Pepsi[a]	46,281	9,188	3,209	6,898	5,400
Keybrd[b]	9,188	3,209	27,290	2,968	3,405
Pepsi	14,796	9,188	3,209	9,188	3,209

Source: Couch, L.W., II, 1997, *Digital and Analog Communication Systems,* 5th ed., Prentice Hall, Upper Saddle River, NJ, pp. 231-232. With permission.
Source: Couch, L.W., II, 1997, *Digital and Analog Communication Systems,* 5th ed., Prentice Hall, Upper Saddle River, NJ, pp. 231-232. With permission.

```
 1  PRINT 0
 2  GO LEFT
 3  GO TO STEP 2 IF 1 IS SCANNED
 4  PRINT 1
 5  GO RIGHT
 6  GO TO STEP 5 IF 1 IS SCANNED
 7  PRINT 1
 8  GO RIGHT
 9  GO TO STEP 1 IF 1 IS SCANNED
10  STOP
```

FIGURE 1.3 The doubling program in the GOTO language. The doubling program in the GOTO language. The doubling program in the GOTO language.

1.2.2 Growth of Academic Programs

Fueling the rapid evolution of curricula in CS&E during the last three decades was an enormous growth in demand, by industry and academia, for computer professionals, researchers, and scientists at all levels. In response, the number of CS&E Ph.D.-granting programs in the U.S. grew from 12 in 1964 to 132 in 1994. During the period 1966–1993, the annual number of bachelor's degrees awarded grew from 89 to 24,580, master's degrees grew from 238 to 10,349, and Ph.D. degrees grew from 19 to 969 [ACM 1968, Andrews 1995].

Figure 1.1 shows the number of bachelor's and master's degrees awarded by U.S. colleges and universities in CS&E from 1966 to 1993. The number of bachelor's degrees peaked at 42,195 in 1986, and then declined, leveling off to a fairly steady 25,000 by 1993. In contrast, master's degree production in computer science has grown steadily throughout the same period and shows no signs of leveling off. The rapid falloff in bachelor's degree production in 1986 may be attributed to the saturation of industry demand for programmers, while the steady growth of master's degrees in recent years may reflect a recognition by industry that an undergraduate computer science degree by itself, while providing good preparation for some positions, is not adequate for many of the newly emerging positions in the technology industry.

Figure 1.2 shows the number of U.S. Ph.D. degrees in computer science and computer engineering during the same 1966–1993 period [Andrews 1995]. The annual number of Ph.D.'s in computer science in the U.S. grew from 19 in 1966 to 878 in 1993, while the overall number of Ph.D.'s in computer science and computer engineering peaked at 1113 in 1992 and leveled off at 969 in 1993 and 1005 in 1994 [Andrews 1995].

Production of M.S. and Ph.D. degrees in computer science and engineering continued to grow into the 1990s, fueled by continuing demand from industry for graduate-level talent and continuing stong demand in academia to staff growing undergraduate and graduate research programs in CS&E. However, there is also a widely held belief that the period of growth in Ph.D. production in CS&E has now leveled off and reached a steady state with respect to demand from industry and academia.

1.2.3 Academic R&D and Growth of Industry Positions

University and industrial research and development (R&D) investments in CS&E grew rapidly in the period 1986–1993. Figure 1.3 shows that academic research and development in computer science nearly doubled, from $321 million to $597 million, during this time period, a growth rate somewhat higher than that of academic R&D in the related fields of electrical engineering and mathematics. During this same period, the overall growth of

academic R&D in engineering and the social sciences also nearly doubled, while that in the physical sciences grew by only about 65%. In 1993, about 68% of the total support for academic R&D came from federal and state sources, while 7% came from industry and 18% came from the institutions themselves [NSF 1995a].

Figure 1.4 shows the growth between 1980 and 1990 in the number of persons with at least a bachelor's degree who were employed in nonacademic (industry and government) computer science positions. Overall, the total number of computer scientists and systems analysts grew by 150%, from 194,100 in 1980 to 485,200 in 1990. In 1990, there were 9400 Ph.D.'s in nonacademic computer science and systems analyst positions (of course, many of these Ph.D.'s were not in computer science) [NSF 1995b]. An informal survey conducted by the Computing Research Association (CRA) suggests that slightly more than half of the domestically employed new Ph.D.'s accepted positions in industry or government in 1994, and the rest accepted academic positions in colleges and universities. This survey also suggests that nearly a third of the total number of 1994 CS&E Ph.D.'s accepted positions abroad [Andrews 1995].

Figure 1.4 also provides some comparative growth information for other professions, again using 1980 and 1990 census data and considering only persons with bachelor's degrees or higher. We see that only the operations and systems researchers' growth of 250% was greater than that of computer scientists, while most professions' growth rates were significantly lower. Overall, the total number of nonacademic scientists and engineers grew from 2,136,200 in 1980 to 3,512,800 in 1990, an increase of 64.4% [NSF 1995b].

1.3 Perspectives in Computer Science and Engineering

Computer science and engineering is a multifaceted discipline that can be viewed from at least four different perspectives. Three of the perspectives—theory, abstraction, and design—underscore the point that computer scientists and engineers approach their work and their subject areas from different intellectual viewpoints. A fourth perspective—the social and professional context—acknowledges that computing directly affects the quality of people's lives, and that computing professionals must be prepared to understand and confront the social issues that arise from their work.

1. The theory of CS&E draws from principles of mathematics and
2. logic as well as from the formal methods of the physical,
3. biological, behavioral, and social sciences. It normally
4. includes the use of advanced mathematical ideas and methods
5. includes the use of advanced mathematical ideas and methods
6. includes the use of advanced mathematical ideas and methods
7. includes the use of advanced mathematical ideas and methods
8. includes the use of advanced mathematical ideas and methods
9. includes the use of advanced mathematical ideas and methods
10. includes the use of advanced mathematical ideas and methods
11. includes the use of advanced mathematical ideas and methods
12. includes the use of advanced mathematical ideas and methods
13. includes the use of advanced mathematical ideas and methods
14. includes the use of advanced mathematical ideas and methods
15. includes the use of advanced mathematical ideas and methods

taken from subfields of mathematics such as algebra, analysis, and statistics. Theory includes the use of various proof and argumentation techniques, like induction and contradiction, to establish properties of formal systems that justify and explain underlying models and paradigms supporting computer science; examples are Church's thesis, the study of algorithmically unsolvable problems, and the study of upper and lower bounds on the complexity of various hard algorithmic problems. Fields like algorithms and to a lesser extent artificial intelligence, computational science, and programming languages have more mature theoretical models than human–computer interaction or graphics, but all ten areas considered here have underlying theories to a greater or lesser extent.

Abstraction in CS&E includes the use of scientific inquiry, modeling, and experimentation to test the validity of hypotheses about computational phenomena. Computer professionals in all ten areas of the discipline use abstraction as a fundamental tool of inquiry—many would argue that computer science is the science of building and examining abstract computational models of reality. Abstraction arises in computer architecture, where the Turing machine serves as an abstract model for complex real computers, and in programming languages, where simple semantic models like the lambda calculus are used as a framework for studying complex languages. It appears in the design of heuristic and approximation algorithms for problems whose optimal solutions are computationally intractable. It is surely used in graphics, where models of 3D objects are constructed mathematically, given properties of lighting, color, and surface texture, and projected in a realistic way on a two-dimensional video screen.

Design is a process used to describe the essential structure of complex systems as a prelude to their implementation. It also encompasses the use of traditional engineering methods, including the classical life-cycle model, to implement efficient and effective computational systems in hardware and software. It includes the use of tools like cost/benefit analysis of alternatives, risk analysis, and fault tolerance that ensure that computing applications are brought to market effectively. Design is a central preoccupation of architects and software engineers developing hardware systems and software applications. Like abstraction, it is an important activity in computational science, database and information retrieval, human–computer interaction, operating systems and networks, and the other areas considered here.

The social and professional context includes many issues that arise at the computer–human interface, such as liability for hardware and software errors, security and privacy of databases and networks, intellectual property issues (patent and copyright), and equity issues (universal access to the technology and the profession). Computing professionals in all subject areas must consider the ethical context in which their work occurs and the special responsibilities that attend their work. The next preliminary chapter discusses these issues, and several other chapters address topics in which specific social and professional issues come into play. For example, security and privacy issues in databases, operating systems, and networks are discussed in Chapters 49 and 89. Risks in software are discussed in several chapters of section X of the Handbook.

1.4 Broader Horizons: HPCC and Grand Challenge Applications

The 1992 report "Computing the Future" (CTF) [CSNRCTB 1992], written by a group of leading computer professionals in response to a request by the Computer Science and Technology Board (CSTB), identifies the need for CS&E to broaden its research agenda and its educational horizons. The view that the research agenda should be broadened initially caused concerns among researchers that funding and other incentives might overemphasize short-term at the expense of long-term goals. This Handbook reflects the broader view

of the discipline in its inclusion of computational science, graphics, and computer–human interaction among the major subfields of computer science.

CTF aimed to bridge the gap between suppliers of research in CS&E and consumers of research such as industry, the Federal government, and funding agencies like NSF, DARPA, and DOE. It addresses fundamental challenges to the field and suggests responses that encourage greater interaction between research and computing practice. Its overall recommendations focus on three priorities:

1. To sustain the core effort that creates the theoretical and experimental science base on which applications build.
2. To broaden the field to reflect the centrality of computing in science and society.
3. To improve education at both the undergraduate and graduate levels.

CTF includes recommendations to federal policy makers and universities regarding research and education:

Recommendations to federal policy makers regarding research: The High-Performance Computing and Communication (HPCC) program passed by Congress in 1989 [OST 1989] should be fully supported.

Recommendations to federal policy makers regarding research: The High-Performance Computing and Communication (HPCC) program passed by Congress in 1989 [OST 1989] should be fully supported.

Recommendations to federal policy makers regarding research: The High-Performance Computing and Communication (HPCC) program passed by Congress in 1989 [OST 1989] should be fully supported.

Recommendations to federal policy makers regarding research: The High-Performance Computing and Communication (HPCC) program passed by Congress in 1989 [OST 1989] should be fully supported.

Though this report was motivated by the desire to provide a rationale for the HPCC program, its message that computer science must be responsive to the needs of society is much broader. The years since publication of CTF have seen a swing away from pure research towards application-oriented research that is reflected in this Handbook. However, it is important to maintain a balance between short-term applications and long-term research in core disciplines.

Though this report was motivated by the desire to provide a rationale for the HPCC program, its message that computer science must be responsive to the needs of society is much broader. The years since publication of CTF have seen a swing away from pure research towards application-oriented research that is reflected in this Handbook. However, it is important to maintain a balance between short-term applications and long-term research in core disciplines.

Though this report was motivated by the desire to provide a rationale for the HPCC program, its message that computer science must be responsive to the needs of society is much broader. The years since publication of CTF have seen a swing away from pure research towards application-oriented research that is reflected in this Handbook. However, it is important to maintain a balance between short-term applications and long-term research in core disciplines.

Though this report was motivated by the desire to provide a rationale for the HPCC program, its message that computer science must be responsive to the needs of society is much broader. The years since publication of CTF have seen a swing away from pure research towards application-oriented research that is reflected in this Handbook. However, it is

important to maintain a balance between short-term applications and long-term research in core disciplines.

Though this report was motivated by the desire to provide a rationale for the HPCC program, its message that computer science must be responsive to the needs of society is much broader. The years since publication of CTF have seen a swing away from pure research towards application-oriented research that is reflected in this Handbook. However, it is important to maintain a balance between short-term applications and long-term research in core disciplines.

Though this report was motivated by the desire to provide a rationale for the HPCC program, its message that computer science must be responsive to the needs of society is much broader. The years since publication of CTF have seen a swing away from pure research towards application-oriented research that is reflected in this Handbook. However, it is important to maintain a balance between short-term applications and long-term research in core disciplines.

Though this report was motivated by the desire to provide a rationale for the HPCC program, its message that computer science must be responsive to the needs of society is much broader. The years since publication of CTF have seen a swing away from pure research towards application-oriented research that is reflected in this Handbook. However, it is important to maintain a balance between short-term applications and long-term research in core disciplines.

The HPCC program encourages universities, research programs, and industry to develop specific capabilities to address the "grand challenges" of the future. Realizing these grand challenges requires both fundamental and applied research, including the development of high-performance computing systems whose speed is two to three orders of magnitude greater than that of current systems, advanced software technology and algorithms that enable scientists and mathematicians to effectively address these grand challenges, networking to support R&D for a gigabit National Research and Educational Network (NREN), and human resources that expand basic research in all areas relevant to high-performance computing.

The grand challenges themselves were identified in HPCC as those fundamental problems in science and engineering with potentially broad economic, political, or scientific impact that can be advanced by applying high-performance computing technology and that can be solved only by high-level collaboration among computer professionals, scientists, and engineers. A list of grand challenges developed by agencies like NSF, DOD, DOE, and NASA in 1989 includes:

- Prediction of weather, climate, and global change.
- Challenges in materials sciences.
- Semiconductor design.
- Superconductivity.
- Structural biology.
- Design of drugs.
- Human genome.
- Quantum chromodynamics.
- Astronomy.
- Transportation.
- Vehicle dynamics and signature.
- Turbulence.

- Nuclear fusion.
- Combustion systems.
- Oil and gas recovery.
- Ocean science.
- Speech.
- Vision.
- Undersea surveillance for antisubmarine warfare.

More recently, an HPCC budget request identifies the following items as the "National Challenges" that face American CS&E and related professions:

- Digital libraries.
- Crisis and emergency management.
- Educational and lifelong learning.
- Electronic commerce.
- Energy management.
- Environmental monitoring and waste management.
- Health care.
- Manufacturing processes and products.
- Public access to government information.

As an outcome of HPCC and CTF, two new subject areas, "computational science" [Stevenson 1994] and "organizational informatics" [Kling 1993] emerged to influence the structure of the original nine subject areas identified in the report "Computing as a Discipline." In this Handbook, we view computational science as an extension of the area of numerical and symbolic computation. This area includes as a central element the fundamental interaction between computation and scientific research. For instance, fields like computational astrophysics, computational fluid dynamics, and computational chemistry all emphasize applications of computing in science and engineering, algorithms, and special considerations for computer architecture. Much of the research and early accomplishments of the emerging computational science field is reported in section IV of this Handbook.

Organizational informatics, on the other hand, emphasizes applications of computing in business and management, information systems and networks, as well as their implementation, risks, and human factors. Some of these intersect in major ways with human–computer interaction, while others fall more directly within the realm of management information systems (MIS), which is usually treated as a separate discipline from computer science and engineering. Thus, in this Handbook, we do not attempt to cover all of organizational informatics as a separate subject area within CS&E. Rather, we include many of its concerns within section VII on Human–Computer Interaction.

In addition, the growth of computer graphics and networks in the last few years provides strong arguments for their inclusion as major subject areas in the discipline. This Handbook distinguishes *graphics* from *human–computer interaction*, of which it had been a subarea in "Computing as a Discipline." Finally, the area of *operating systems and networks* in this Handbook has evolved out of what had been called *operating systems* in "Computing as a Discipline," in recognition of the rapid growth of distributed computing in the last few years.

1.5 Organization and Content

In the 1940s computing was identified with number crunching, and numerical analysis was considered a central tool. Hardware, logical design, and information theory emerged as important subfields in the early 1950s. Software and programming emerged as important subfields in the mid 1950s and soon dominated hardware as topics of study in computer science. In the 1960s computer science could be comfortably classified into theory, systems (including hardware and software), and applications. Software engineering emerged as an important subdiscipline in the late 1960s. The 1980 Computer Science and Engineering Research Study [Arden 1980] classified the discipline into nine subfields:

- Numerical computation.
- Theory of computation.
- Hardware systems.
- Artificial intelligence.
- Programming languages.
- Operating systems.
- Database management systems.
- Software methodology.
- Applications.

This Handbook's classification into ten subfields is quite similar to that of the COSERS study, suggesting that the discipline of CS&E is stabilizing:

- Algorithms and data structures.
- Architecture.
- Artificial intelligence.
- Computational science.
- Database and information retrieval.
- Graphics.
- Human–computer interaction.
- Operating systems and networks.
- Programming languages.
- Software engineering.

This Handbook's classification has discarded numerical analysis and added the new areas of human–computer interaction and graphics.

> "A Process is a structured, measured set of activities designed to produce a specific output for a particular customer or market—A process is thus a specific ordering of work activities across time and space, with a beginning, an end. and clearly defined inputs and outputs: a structure for action."
>
> Thomas Davenport
> Senior Adjutant to the Junior Marketing VP

The other eight areas appear in both classifications with some name changes (theory of computation has become algorithms and data structures, applications has become compu-

tational science, hardware systems has become architecture, operating systems has added networks, and database has added information retrieval as important new directions).

Though the high-level classification has remained stable, the content of each area has evolved and matured. We examine below the scope of each area and the topics within each area treated in the Handbook.

1.5.1 Algorithms and Data Structures

The subfield of algorithms and data structures is interpreted broadly to include core topics in the theory of computation as well as data structures and practical algorithm techniques. Its thirteen chapters provide a comprehensive overview that spans both theoretical and applied topics in the analysis of algorithms. Chapter 3 introduces fundamental concepts such as computability and undecidability and formal models such as Turing machines and Chomsky grammars, while Chapter 4 reviews techniques of algorithm design like divide and conquer, dynamic programming, recurrence relations, and greedy heuristics.

Chapter 5 covers data structures both descriptively and in terms of their space–time complexity, while Chapter 6 reviews methods and techniques of computational geometry, and Chapter 7 presents the rich area of randomized objects. Pattern matching and text compression algorithms are examined in Chapter 8, graph and network algorithms in Chapter 9, and algebraic algorithms in Chapter 10. Chapter 11 examines topics in complexity like P vs. NP, NP-completeness, and circuit complexity, while Chapter 12 examines parallel algorithms, and Chapter 13 considers combinatorial optimization. Chapter 14 concludes section I with a case study in VLSI layout that makes use of partitioning, divide and conquer, and other algorithm techniques common in VLSI design.

1.5.2 Architecture

Computer architecture is the design of efficient and effective computer hardware at all levels, from the most fundamental concerns of logic and circuit design to the broadest concerns of parallelism and high-performance computing. The chapters in section II span all these levels, providing a sampling of the principles, accomplishments, and applications of modern computer architectures.

Chapters 15 and 16 introduce the fundamentals of logic design components, including elementary circuits, Karnaugh maps, programmable array logic, circuit complexity and minimization issues, arithmetic processes, and speedup techniques. The architecture of buses is covered in Chapter 17, while the principles of memory architecture are addressed in Chapter 18. Topics there include associative memories, cache design, interleaving, and memories for pipelined and vector processors.

Chapter 19 concerns the design of effective and efficient computer arithmetic units. Chapter 20 extends the design horizon by considering the various models of parallel architectures, including the performance of contemporary machines that fall into the SIMD, MISD, and MIMD categories.

1.5.3 Artificial Intelligence and Robotics

Artificial intelligence (AI) is the study of the computations that make it possible to simulate human perception, reasoning, and action. Current efforts are aimed at constructing computational mechanisms that process visual data, understand speech and written language, control robot motion, and model physical and cognitive processes. Robotics is a complex field, drawing heavily from AI as well as other areas of science and engineering.

AI includes techniques for automated learning, planning, and representing knowledge.

Chapter 21 opens this section with a discussion of deductive learning. The use of decision trees and neural networks in learning and other areas is the subject of Chapters 22 and 23, while Chapter 24 introduces genetic algorithms.

Chapter 25 focuses on the area of computer vision. Chapter 26 addresses issues related to the mechanical understanding of spoken language. Chapter 27 presents the rationale and uses of planning and scheduling models in AI research. Chapter 28 describes the principles of knowledge representation and their applications in natural-language processing (NLP).

Artificial-intelligence work requires a number of distinct tools and models. These include the use of fuzzy, temporal, and other logics, as described in Chapter 29. The use of a variety of specialized search techniques to address the combinatorial explosion of alternatives in many AI problems is the subject of Chapter 30. Many AI applications must handle the notion of uncertainty. Chapter 31 discusses the modeling of decision making under uncertainty, while the related idea of qualitative modeling is discussed in Chapter 32.

Chapter 33 concludes section III with a thorough discussion of the principles and major results in the field of robotics: the design of effective devices that simulate mechanical, sensory, and intellectual functions of humans in specific task domains such as factory production lines.

1.5.4 Computational Science

The emerging area of computational science unites computational simulation, experimental investigations, and theoretical pursuits as three fundamental modes of scientific discovery. It uses scientific visualization, made possible by computational simulation, as a window into the analysis of physical phenomena and processes, providing a virtual microscope/telescope for inquiry and investigation at an unprecedented level of detail.

Section IV focuses on the challenges and opportunities offered by computers in aiding scientific analysis and engineering design. Chapter 34 introduces the section by presenting the fundamental subjects of computational geometry and grid generation. The design of graphical models for scientific visualization of complex physical and biological phenomena is the subject of Chapter 35.

Each of the remaining chapters in this section covers the computational science challenges and discoveries in a particular scientific or engineering field. Chapter 36 presents the computational aspects of structural mechanics, while Chapter 37 does the same for fluid dynamics. Computational reacting flow is the subject of Chapter 38, while Chapter 39 summarizes the progress in the area of computational electromagnetics. Chapter 40 addresses the grand challenge of computational ocean modeling. This section closes with a discussion of computational biological modeling in Chapter 41.

1.5.5 Database and Information Retrieval

The subject area of database and information retrieval addresses the general problem of storing large amounts of data in such a way that they are reliable, up to date, and efficiently retrieved. This problem is prominent in a wide range of applications in industry, government, and academic research. Availability of such data on the Internet and in forms other than text (e.g., audio and video) makes this problem increasingly complex.

At the foundation are the fundamental data models (relational, hierarchical, entity–relationship, network, and object-oriented) discussed in Chapter 42. The conceptual, logical, and physical levels of designing a database for high performance in a particular application domain are discussed in Chapter 43.performance in a particular application domain are discussed in Chapter 43.

A number of basic issues surround the design of database models and systems. These include choosing from alternative access methods (Chapter 44), optimizing database queries (Chapter 45), controlling concurrency (Chapter 46), and benchmarking database workloads and performance (Chapter 47).

The design of heterogeneous databases and interoperability is discussed in Chapter 48. The issue of database security and privacy protection, in stand-alone and networked environments, is the subject of Chapter 49. The special considerations involved in storing and retrieving information from text databases are covered in Chapter 50.

A special topic in database research is the study of deductive (rule-based) databases, addressed in Chapter 51. Chapter 53 closes section V with a case study on SQL, a widely used database query language standard.

1.5.6 Graphics

Computer graphics is the study and realization of complex processes for representing physical and conceptual objects visually on a computer screen. These processes include the internal modeling of objects, rendering, hidden-surface elimination, color, shading, projection, and representing motion. An overview of these processes and their interaction is presented in Chapter 54.

Fundamental to all graphics applications are the processes of modeling and rendering. Modeling is the design of an effective and efficient internal representation for geometric objects (points, lines, polygons, solids, fractals, and their transformations), which is the subject of Chapters 55 and 56. Rendering, the process of representing the objects in a three-dimensional scene on a two-dimensional screen, is discussed in Chapter 57. Among its special challenges are the elimination of hidden surfaces, color, illumination, and shading.

The reconstruction of scanned images is another important area of computer graphics. Sampling, filtering, reconstruction, and antialiasing are the focus of Chapter 58. The representation and control of motion, or animation, is another complex and important area of computer graphics. Its special challenges are presented in Chapter 59.

Chapter 60 discusses volume data sets, and Chapter 61 looks at the emerging field of virtual reality and its particular challenges for computer graphics. Chapter 62 concludes section VI with a discussion of Renderman as a case study of a particularly effective application of the principles of computer graphics in the real world.

1.5.7 Human–Computer Interaction

This area, the study of how humans and computers interact, has the goal of improving the quality of the interaction and the effectiveness of those who use technology. This includes the conception, design, implementation, risk analysis, and effects of user interfaces and tools on those who use them in their work.

Chapter 63 opens section VII with a discussion of methods of overall system modeling, including users and modes of use. Modeling the organizational environments in which technology users work is the subject of Chapter 64. Usability engineering is the focus of Chapter 65, while user interface design methods are discussed in Chapter 66. The impact of international standards for user interfaces on the design process is the main concern of Chapter 67.

Specific devices, tools, and techniques for effective user-interface design form the basis for the next few chapters in this section. Chapters 68 and 69 discuss, respectively, the characteristics of input devices like the mouse and keyboard and output devices like computer screens and multimedia audio devices. Chapter 70 focuses on design techniques for effective

interaction with users through these devices. The special concerns for integrating multimedia with user interaction are presented in Chapter 71. Lower-level concerns for the design of interface software technology are addressed in Chapter 72. The programming and software development process for user-interface implementation is discussed in Chapter 73. The effective presentation of documentation, training, and help facilities for users is a perennial concern for software designers, and its current status is reviewed in Chapter 74.

1.5.8 Operating Systems and Networks

Operating systems form the software interface between the computer and its applications. Section VIII covers their analysis and design, their performance, and their special challenges in a networked computing environment. Chapter 75 briefly traces the historical development of operating systems and introduces the fundamental terminology, including process scheduling, memory management, synchronization, I/O management, and distributed systems.

Think About It...

Commonly thought of as the first modern computer, ENTAC was built in 1944. It took up more space than an 18-wheeler's tractor trailer and weighed more than 17 Chevrolet Camaros. It consumed 140,000 watts of electricity while executing up to 5,000 basic arithmetic operations per second. One of today's popular microprocessors, the 486, is built on a tiny piece of silicon about the size of a dime.

With the continual expansion of capabilities, computing power will eventually exceed the capacity for human comprehension or human control.

The Information Revolution

Business Week

The process is a key unit of abstraction in operating-system design. Chapter 76 discusses the dynamics of processes and threads. Strategies for process and device scheduling are presented in Chapter 77. The special requirements for operating systems in real-time and embedded system environments are the subject of Chapter 78. Algorithms and techniques for process synchronization and interprocess communication are the subject of Chapter 79.

Memory and input/output device management is also a central concern of operating systems. Chapter 80 discusses the concept of virtual memory, from its early incarnations to its uses in present-day systems and networks. The different types and access methods for secondary storage and file systems are covered in Chapter 81.

Box Title

Extending operating system functionality across a networked environment adds another level of complexity to the design process. Chapter 82 presents an overview of network organization and topologies, while Chapter 83 describes network routing protocols. The topology and functionality of internetworking, with the Internet as prime example, are presented in Chapter 84.

The influence of networked environments on the design of distributed operating systems is considered in Chapter 85. Distributed file and memory systems are discussed in Chapter 86, while distributed and multiprocessor scheduling are the focus of attention in Chapter 87. Finally, the forward-looking notion of dynamically partitioning a computing task across a network of heterogeneous computers is the topic of Chapter 88.

Operating systems and networks, especially the Internet, must make provisions for ensuring system integrity in the event of inappropriate access, unexpected malfunction and breakdown, and violations of security or privacy principles. Chapter 89 introduces some of the security and privacy issues that arise in a networked environment. Models for system security and protection are the subject of Chapter 90, while Chapter 91 discusses authentication, access control, and intrusion detection. Chapter 92 focuses on security issues that arise in networks, while a case discussion of some noteworthy malicious software and hacking events appears in Chapter 93.

1.5.9 Programming Languages

In section IX the design space of programming languages is partitioned into paradigms, mechanisms for compiling, and run-time management, and the theoretical areas of foundational models, type systems, and semantics are examined. Overall, this section provides a good balance between considerations of language paradigms, implementation issues, and

Chapter 94 considers traditional language and implementation questions for imperative programming languages like Fortran, C, Pascal, and Ada 83. Chapter 95 considers topics in functional programming like lazy and eager evaluation, and Chapter 96 examines object-oriented concepts like classes, inheritance, encapsulation, and polymorphism. Chapter 97 considers declarative programming in the logic/constraint programming paradigm, while Chapter 98 considers issues in concurrent/distributed programming as well as parallel models of computation. Compilers and interpreters for sequential languages are considered in Chapter 99, while compilers for parallel architectures and dataflow languages are considered in Chapter 100. The issues surrounding run-time environments and memory management for compilers and interpreters are addressed in Chapter 101.

1.5.10 Software Engineering

Section X on software engineering examines formal specification, design, verification and testing, project management, and other aspects of the software life cycle. Chapter 105 considers models of the software life cycle such as the waterfall and spiral models as well as specific phases of the life cycle. Chapter 106 examines software qualities like maintainability, portability, and reuse that are needed for high-quality software systems, while Chapter 107 considers formal models, specification languages, and the specification process.

Chapter 108 deals with the traditional and object-oriented design processes, featuring a case study in top-down functional design. Chapter 109 on verification and validation deals with the use of systematic techniques like verification and testing for quality assurance, while Chapter 110 examines testing models as well as risk and reliability issues.

Chapter 111 considers methods of project design such as chief programmer teams and rapid prototyping, as well as project scheduling and evaluation. Chapter 112 considers software tools like compilers, editors, and CASE tools and surveys graphical environments. Chapter 113 on interoperability considers architectures for communicating among heterogeneous software components such as OMG's Common Object Request Broker Architecture (CORBA) and Microsoft's Common Object Model (COM).

Glossary

Term 1 This is definition of Term 1. This is definition of Term 1. This is definition of Term 1. This is definition of Term 1.

Term 2 This is definition of Term 2. This is definition of Term 2. This is definition of Term 2. This is definition of Term 2.

Term 3 This is definition of Term 3. This is definition of Term 3. This is definition of Term 3. This is definition of Term 3.

Term 4 This is definition of Term 4. This is definition of Term 4. This is definition of Term 4. This is definition of Term 4.

Term 5 This is definition of Term 5. This is definition of Term 5. This is definition of Term 5. This is definition of Term 5.

Term 6 This is definition of Term 6. This is definition of Term 6. This is definition of Term 6. This is definition of Term 6.

Term 7 This is definition of Term 7. This is definition of Term 7. This is definition of Term 7. This is definition of Term 7.

1.6 Conclusion

In 1997, the ACM celebrates its 50th anniversary. The first 50 years of CS&E are characterized by dramatic growth and evolution. While it is safe to affirm today that the field has reached a certain level of maturity, it would be foolish to assume that it will remain unchanged in the future. Already, conferences are calling for new visions that will enable the discipline to continue its rapid evolution into the twenty-first century. This Handbook is designed to convey the modern spirit, accomplishments, and direction of CS&E as we see it in 1996. It interweaves theory with practice, highlighting "best practices" in the field as well as current research directions. It provides today's answers to well-formed questions posed by professionals and researchers across the ten major subject areas. Finally, it identifies key professional and social issues that lie at the intersection of the technical aspects of CS&E and its impact in the world.

The future holds great promise for the next generations of computer scientists and

engineers. These people will solve problems that have only recently been conceived, such as those suggested by the HPCC as "grand challenges." To address these problems, and to extend these solutions in a way that benefits the lives of significant numbers of the world's population, will require substantial energy, commitment, and real investment on the part of institutions and professionals throughout the world. The challenges are complex, and the solutions are not likely to be obvious.

2

Ethical Issues for Computer Scientists and Engineers

Author Second Chapter
Author Affiliation

Author Chapter
Second Affiliation

2.1 Introduction: Why a Chapter on Ethical Issues? This head can be a lenghty one

Computers have had a powerful impact on our world and are destined to shape our future. This observation, [**?**] now commonplace, is the starting place for any discussion of professionalism and ethics in computing. The work of computer scientists and engineers is part of the social, political, economic, and cultural world in which we live, and affects many aspects of that world. Professionals who work with computers have special knowledge and that knowledge, when combined with computers, has significant power to change people's lives.

In this chapter of the Handbook we provide a perspective on the role of computer and engineering professionals and we examine the relationships and responsibilities that go with having and using computing expertise. In addition to the topic of professional ethics, we briefly discuss several of the social-ethical issues created or exacerbated by increasing use of computers: privacy, property, risk and reliability, and global communication.

Computers, digital data, and telecommunications have [**?**] changed work, travel, education, business, entertainment, government, and manufacturing. For example, work now increasingly involves sitting in front of a computer screen and using a keyboard to make things happen in a manufacturing process or to keep track of records where in the past these same tasks would have involved physically lifting, pushing, and twisting or using pens, paper, and file cabinets. Changes such as these—in the way we do things—have, in

```
 1  PRINT 0
 2  GO LEFT
 3  GO TO STEP 2 IF 1 IS SCANNED
 4  PRINT 1
 5  GO RIGHT
 6  GO TO STEP 5 IF 1 IS SCANNED
 7  PRINT 1
 8  GO RIGHT
 9  GO TO STEP 1 IF 1 IS SCANNED
10  STOP
```

FIGURE 2.1 The doubling program in the GOTO language. The doubling program in the GOTO language. The doubling program in the GOTO language.

turn, fundamentally changed who we are as individuals, communities, and nations. Some would argue, for example, that new kinds of communities (e.g., cyberspace on the Internet) are forming, individuals are developing new types of personal identity, and new forms of authority and control are taking hold as a result of this evolving technology.

Computer technology is shaped by social-cultural concepts, laws, the economy, and politics [**?**]. These same concepts, laws, and institutions have been pressured, challenged, and modified by computer technology. Technological advances can antiquate laws, concepts, and traditions, compelling us to reinterpret and create new laws, concepts, and moral notions [**?**]. Our attitudes about work and play, our values, and our laws and customs are deeply involved in technological change.

When it comes to the social-ethical issues surrounding computers, some have argued that the issues are not unique. All of the ethical issues raised by computer technology can, it is said, be classified and worked out using traditional moral concepts, distinctions, and theories. There is nothing new here in the sense that all of the issues have to do with traditional moral concepts such as privacy, property, responsibility, and traditional moral ends such as maximizing individual freedom or holding individuals accountable. These concepts and values predate computers; hence, it would seem there is nothing unique about *computer ethics*.

On the other hand, those who argue for the uniqueness of the topic point to the fundamental ways that computers have changed so many human activities, such as manufacturing, record keeping, banking, and communicating. The change is so radical, it is claimed, that traditional moral concepts, distinctions, and theories, if not abandoned, must be significantly reinterpreted and extended. For example, they must be extended to computer-mediated relationships, computer software, computer art, electronic bulletin boards, and so on. The


```
 1  PRINT 0
 2  GO LEFT
 3  GO TO STEP 2 IF 1 IS SCANNED
 4  PRINT 1
 5  GO RIGHT
 6  GO TO STEP 5 IF 1 IS SCANNED
 7  PRINT 1
 8  GO RIGHT
 9  GO TO STEP 1 IF 1 IS SCANNED
10  STOP
```


FIGURE 2.2 The doubling program in the GOTO language. The doubling program in the GOTO language. The doubling program in the GOTO language.

```
 1  PRINT 0
 2  GO LEFT
 3  GO TO STEP 2 IF 1 IS SCANNED
 4  PRINT 1
 5  GO RIGHT
 6  GO TO STEP 5 IF 1 IS SCANNED
 7  PRINT 1
 8  GO RIGHT
 9  GO TO STEP 1 IF 1 IS SCANNED
10  STOP
```

FIGURE 2.3 The doubling program in the GOTO language. The doubling program in the GOTO language. The doubling program in the GOTO language.

uniqueness of the ethical issues surrounding computers can be argued for in a number of ways. Computer technology makes possible a scale of activities not possible before. This includes a larger scale of record keeping of personal information, as well as larger scale calculations which, in turn, allow us to build and do things not possible before, such as undertaking space travel and operating a global communication system. In addition to scale, computer technology has involved the creation of new kinds of entities for which no rules initially existed: entities such as computer files, computer programs, and user interfaces. The uniqueness argument can also be made in terms of the power and pervasiveness of computer technology. It seems to be bringing about a magnitude of change comparable to that which took place during the Industrial Revolution, transforming our social, economic, and political institutions, our understanding of what it means to be human, and the distribution of power in the world. Hence, it would seem the issues are special if not unique.

A synthesis of these two views of computer ethics seems necessary since analysis of a computer ethical issue generally involves both working on something new and drawing on something old. Issues in computer ethics are new species of older ethical problems [Johnson 1994]. Most of the issues can be understood using traditional moral concepts such as autonomy, privacy, property, and responsibility. Most arise in contexts in which there are already social, ethical, and legal norms; that is, the issues arise in the context of the workplace, government, business, role relationships, and so on. In this respect, the issues are not new or unique. Nevertheless, when a computer is involved, the situation may have special features which have not been addressed by prevailing norms, and these features make a moral difference. For example, although property rights and even intellectual property rights have been worked out in the past, software has raised new property issues: Should the arrangement of icons appearing on the screen of a user interface be ownable? Is there anything intrinsically wrong with copying software? Software has features which make the distinction between idea and expression (a distinction at the core of copyright law) almost incoherent. As well, it has features which make standard intellectual property laws difficult to enforce. Hence, questions about what should be owned when it comes to software and how to evaluate violations of software ownership rights are not new in the sense that they are property rights issues, but they are new in the sense that nothing with the characteristics of software has been addressed before. We have, then, a new species of traditional property rights.

Similarly, although our understanding of rights and responsibilities in the employer–employee relationship have been evolving for centuries, never before have employers had the capacity to monitor their workers electronically, keeping track of every keystroke, and recording and reviewing all work done by an employee (covertly or with prior consent). When we evaluate this new monitoring capability and ask whether employers should use it, we are working on an issue that has never arisen before, though many other issues involving employer–employee rights have. We address a new species of the tension between

employer–employee rights and interests.

The social-ethical issues posed by computer technology are significant in their own right, but they are of special interest here because computer and engineering professionals bear responsibility for this technology. It is of critical importance that they understand the social change brought about by their work and the difficult social-ethical issues posed. Just as some have argued that the social-ethical issues posed by computer technology are not unique, some have argued that the issues of professional ethics surrounding computers are not unique. We propose, in parallel with our previous genes-species argument, that the professional ethics issues arising for computer scientists and engineers are species of generic issues of professional ethics. All professionals have responsibilities to their employers, clients, coprofessionals, and to the public. Managing these types of responsibilities poses a challenge in all professions. Moreover, all professionals bear some responsibility for the impact of their work. In this sense, the professional ethics issues arising for computer scientists and engineers are generally similar to those in other professions. Nevertheless, it is also true to say that the issues arise in unique ways for computer scientists and engineers because of the special features of computer technology.

In what follows, we discuss ethics in general, professional ethics, and finally, the ethical issues surrounding computer technology.

2.2 Ethics in General

There is a lively history of ethical theories. Ethicists explore theories to: (1) explain and justify prevailing moral notions, (2) critique ordinary moral beliefs, and (3) assist in rational, ethical decision making. It is not our purpose here to propose, defend, or attack any particular ethical theory. Rather, we offer brief descriptions of three theories to illustrate the nature of ethical analysis. We also include a decision making method that combines elements of each theory.

Ethical analysis involves giving reasons for moral claims and commitments. It is not just a matter of articulating intuitions. When the reasons given for a claim are developed into a moral theory, the theory can be incorporated into techniques for improved technical decision making. Three traditions in ethical analysis and problem solving are described. This is by no means an exhaustive account, nor is our description of any of the three any more than a brief introduction. The three traditions are utilitarianism, deontology, and social contract theory.

2.2.1 Utilitarianism

Utilitarianism has greatly influenced 20th-century thinking. According to this theory, we should make decisions about what to do by focusing on the consequences of our actions. Ethical rules are derived from their usefulness (their utility) in bringing about happiness. Utilitarianism offers one seemingly simple moral principle which everyone should use to determine what to do in a given situation: everyone ought to act so as to bring about the greatest amount of happiness for the greatest number of people.

According to utilitarianism, happiness is the only value that can serve as the fundamental base for ethics. Since happiness is the ultimate good, morality must be based on creating as much of this good as possible. The utilitarian principle provides a decision procedure. When you want to decide what to do, the alternative that produces the most overall net happiness (good minus bad) is the right action. The right action may be one that brings

about some unhappiness but that is justified if the action also brings about enough happiness to counterbalance the unhappiness, or if the action brings about the least unhappiness of all possible alternatives.

Be careful not to confuse utilitarianism with egoism. Egoism is a theory claiming that one should act so as to bring about the most good consequences for one's self. Utilitarianism does not say that you should maximize your own good. Rather, total happiness in the world is what is at issue; when you evaluate your alternatives you have to ask about their effects on the happiness of everyone. It may turn out to be right for you to do something that will diminish your own happiness because it will bring about an increase in overall happiness.

The emphasis on consequences found in utilitarianism is very much a part of decision making in our society, in particular as a framework for law and public policy. Cost-benefit and risk-benefit forms of analysis are, for example, consequentialist in character.

Utilitarians do not all agree on the details of utilitarianism; there are different kinds of utilitarians. One issue is whether the focus should be on *rules* of behavior or individual *acts*. Utilitarians have recognized that it would be counter to overall happiness if each one of us had to calculate at every moment what the consequences of every one of our actions would be. Sometimes we must act quickly, and often the consequences are difficult or impossible to foresee. Thus, there is a need for general rules to guide our actions in ordinary situations. Hence, *rule utilitarians* argue that we ought to adopt rules which, if followed by everyone, would, in general and in the long run, maximize happiness. *Act utilitarians*, on the other hand, put the emphasis on judging individual actions rather than creating rules.

Both rule utilitarians and act utilitarians share an emphasis on consequences; deontological theories do not share this emphasis.

2.2.2 Deontological Theories

Deontological theories can be understood as a response to the criticisms of utilitarian theories. A traditional criticism of utilitarianism is that it sometimes leads to conclusions that are incompatible with our most strongly held moral intuitions. Utilitarianism seems, for example, open to the possibility of justifying enormous burdens on some individuals for the sake of others. To be sure, every person counts equally; no one person's happiness or unhappiness is more important than any other person's. However, since utilitarians are concerned with the total amount of happiness, we can imagine situations where great overall happiness might result from sacrificing the happiness of a few. Suppose, for example, that having a small number of slaves would create great happiness for large numbers of people; or suppose we kill one healthy person and use the resulting body parts to save ten people in need of transplants.

Critics of utilitarianism say that if utilitarianism justifies such practices, then the theory must be wrong. Utilitarians have a defense, arguing that such practices could not be justified in utilitarianism because of the long-term consequences. Such practices would produce so much fear that the happiness temporarily created would never counterbalance the unhappiness of everyone living in fear that they might be taken for sacrifice.

We need not debate utilitarianism here. The point is that deontologists find utilitarianism problematic because it puts the emphasis on the consequences of an act rather than on the act itself. Deontological theories claim that the internal character of the act is what is important. The rightness or wrongness of an action depends on the principles inherent in the action. If an action is done from a sense of duty, and if the principle of the action can be universalized, then the action is right. For example, if I tell the truth because it is convenient for me to do so, or because I fear the consequences of getting caught in a lie, my action is not worthy. A worthy action is an action that is done from duty, which involves

respecting other people, recognizing them as ends in themselves, not as means to some good effect.

According to deontologists, utilitarianism is wrong because it treats individuals as means to an end (maximum happiness). For deontologists, what grounds morality is not happiness, but human beings as rational agents. Human beings are capable of reasoning about what they want to do. The laws of nature determine most activities: plants grow towards the sun, water boils at a certain temperature, and objects fall at a constant rate in a vacuum. Human action is different in that it is self-determining; humans initiate action after thinking, reasoning, and deciding. The human capacity for rational decisions makes morality possible, and it grounds deontological theory. Because each human being has this capacity, each human being must be treated accordingly: with respect. No one else can make our moral choices for us, and each of us must recognize this capacity in others.

Although deontological theories can be formulated in a number of ways, one formulation is particularly important: Immanuel Kant's categorical imperative [Kant 1785]. There are three versions of it, and the second version goes as follows: *Never treat another human being merely as a means but always as an end.* It is important to note the *merely* in the categorical imperative. Deontologists do not insist that we never use another person; only that we never *merely* use them. For example, if I own a company and hire employees to work in my company, I might be thought of as using those employees as a means to my end (i.e., the success of my business). This, however, is not wrong if the employees agree to work for me and if I pay them a fair wage. I thereby respect their ability to choose for themselves and I respect the value of their labor. What would be wrong would be to take them as slaves and make them work for me, or to pay them so little that they must borrow from me and must remain always in my debt. This would show disregard for the value of each person as a freely choosing, rationally valuing, specially efficacious person.

2.2.3 Social Contract Theories

A third tradition in ethics thinks of ethics on the model of a social contract. There are many different social contract theories, and some, at least, are based on a deontological principle. Individuals are rational free agents; hence, it is immoral to exert undue power over them, to coerce them. Government and society are problematic insofar as they seem to force individuals to obey rules, apparently treating individuals as means to social good. Social contract theories get around this problem by claiming that morality (and government policy) are, in effect, the outcome of rational agents agreeing to social rules. In agreeing to live by certain rules, we make a contract. Morality and government are not, then, systems imposed on individuals; they do not exactly involve coercion. Rather, they are systems created by freely choosing individuals (or they are institutions that rational individuals would choose if given the opportunity).

Philosophers such as Rousseau, Locke, Hobbes, and more recently Rawls [1971] are generally considered social contract theorists. They differ in how they get to the social contract and what it implies. For our purposes, however, the key idea is that principles and rules guiding behavior may be derived from identifying what it is that rational (even self-interested) individuals would agree to in making a social contract. Such principles and rules are the basis of a shared morality. For example, it would be rational for me to agree to live by rules that forbid killing and lying. Even though such rules constrain me, they also give me some degree of protection: if they are followed, I will not be killed or lied to.

The social contract theory cannot be used simply by asking what rules you would agree to now. Most theorists recognize that what you would agree to now is influenced by your present position in society. Most individuals would opt for rules that would benefit their

particular situation and characteristics. Hence, most social contract theorists insist that the principles or rules of the contract must be derived by assuming certain things about human nature or the human condition. Rawls, for example, insists that we imagine ourselves behind a *veil of ignorance*. We are not allowed to know important features about ourselves, e.g., what talents we have, what race, gender we will be, for if we know these things we will not agree to just rules, but only rules that will maximize our self-interest. Justice consists of the rules we would agree to when we do not know who we are, for we would want rules that would give us a fair situation no matter where we ended up in the society.

2.2.4 A Paramedic Method for Computer Ethics

Drawing on elements of the three theories described, Collins and Miller [1992] have proposed a decision assisting method, called the paramedic method for computer ethics. This is not an algorithm for solving ethical problems; it is not nearly detailed or objective enough for that designation. It is merely a guideline for an organized approach to ethical problem solving.

Assume that a computer professional is faced with a decision that involves human values in a significant way. There may already be some obvious alternatives, and there also may be creative solutions not already discovered. The paramedic method is designed to help the professional analyze alternative actions and to encourage the development of creative solutions. The method proceeds as follows:

1. Identify alternative actions; list the few alternatives that seem most promising. If an action requires a long description, summarize it as a title with just a few words. Call the actions $A1$, $A2$, ..., Aa. No more than five actions should be analyzed at a time.
2. Identify people, groups of people, or organizations that will be affected by the decision that must be made. Again, hold down the number of entities to the five or six that are affected most. Label the people $P1$, $P2$, ..., P_p.
3. Make a table with the horizontal rows labeled by the identified people and the vertical columns labeled with the identified actions. We call such a table a $P \times A$ table. Make two copies of the $P \times A$ table, and label one as the *opportunities* table and the other as the *vulnerabilities* table. In the opportunities table, list in each interior cell of the table at entry $[x, y]$ the possible good that is likely to happen to person x if action y is taken. Similarly, in the vulnerability table, at position $[x, y]$ list all of the things that are likely to happen badly for x if the action y is taken. These two graphs represent benefit/cost calculations for a consequentialist, utilitarian analysis.
4. Make a new table with the set of persons marking both the columns and the rows (a $P \times P$ table). In each cell $[x, y]$ name any responsibilities or duties that x owes y in this situation. (The cells on the diagonal $[x, x]$ are important; they list things one owes oneself.) Now make copies of this table, labeling one copy for each of the alternative actions being considered. Work through each cell $[x, y]$ of each table and place a $+$ next to a duty if the action for that table is likely to fulfill the duty x owes y; mark the duty with a $-$ if the action is unlikely to fulfill that duty; mark the duty with a $+/-$ if the action partially fulfills it and partially does not; and mark the duty with a ? if the action is irrelevant to the duty or if it is impossible to predict whether or not the duty will be fulfilled. (Few cells generally fall into this last category.)
5. Review the tables from steps 3 and 4. Envision a meeting of all of the parties (or one representative from each of the groups) in which no one knows which role they will take or when they will leave the negotiation. Which alternative do you

think such a group would adopt, if any? Do you think such a group could discover a new alternative, perhaps combining the best elements of the previously listed actions? If this thought experiment produces a new alternative, expand the $P \times A$ tables from step 3 to include the new alternative action, and make a new copy of the $P \times P$ table in step 4 and do the + and − marking for the new table.

6. If any one of the alternatives seems to be clearly preferred (i.e., it has high opportunity and low vulnerability for all parties, and tends to fulfill all the duties in the $P \times P$ table), then that becomes the recommended decision. If no one alternative action stands out, the professionals can examine tradeoffs using the charts, or can iteratively attempt step 5 (perhaps with outside consultations) until an acceptable alternative is generated.

Using the paramedic method can be time consuming, and it does not eliminate the need for judgement. But it can help organize and focus analysis as an individual or group works through the details of a case situation to arrive at a decision.

2.2.5 Easy and Hard Ethical Decision Making

Sometimes ethical decision making is easy; for example, when it is clear that an action will prevent a serious harm and has no drawbacks, then that action is the ethical thing to do. Sometimes, however, ethical decision making is more complicated and challenging. Take the following case: your job is to make decisions about which parts to buy for a computer manufacturing company; a person who sells parts to the company offers you tickets to an expensive Broadway show; should you accept the tickets? In this case the right thing to do is more complicated because you may be able to accept the tickets and not have this affect your decision about parts. You owe your employer a decision on parts that is in the best interests of the company, but will accepting the tickets influence future decisions? Other times you know what the right thing to do is but doing it will have such great personal costs, that you can not bring yourself to do it; for example, you might be considering blowing the whistle on your employer who has been extremely kind and generous with you, but who now has asked you to cheat on the testing results on a life-critical software system designed for a client.

To make good decisions, professionals must be aware of potential issues and must have a fairly clear sense of their responsibilities in various kinds of situations. This often requires sorting out complex relationships and obligations, anticipating the effects of various actions, and balancing responsibilities to multiple parties. This activity is part of professional ethics.

2.3 Professional Ethics

Ethics is not just a matter for individuals as individuals. We all occupy a variety of social roles which carry with them special responsibilities and privileges. As parents, we have special responsibilities for children. As citizens, members of churches, officials in clubs, and so on, we have special rights and duties and so it is with professional roles. Being a professional is often distinguished from merely having an occupation because a professional makes a different sort of commitment. Being a professional means more than just having a job. The difference is commitment to doing the right thing because you are a member of a group that has taken on responsibility for a domain of social activity—a social function. The group is accountable to society for this domain, and for this reason, professionals must behave in ways that are worthy of public trust.

Some theorists explain this commitment in terms of a social contract between a profession and the society in which it functions. Society grants special rights and privileges to the professional group, such as control of admission to the group, access to educational

institutions, and confidentiality in professional–client relationships. Society, in turn, may even grant the group a monopoly over a domain of activity (e.g., only licensed engineers can sign off on construction designs, only doctors can prescribe drugs). In exchange, the professional group promises to self-regulate and practice its profession in ways that are beneficial to society, i.e., to promote safety, health, welfare. The social contract idea is a way of illustrating the importance of the trust that clients and the public put in professionals; it shows the importance of professionals acting so as to be worthy of that trust.

The special responsibilities of professionals have been accounted for in other theoretical frameworks as well. Davis [1995], for example, argues that members of professions implicitly, if not explicitly, agree among themselves to adhere to certain standards because this elevates the level of activity. If all computer scientists and engineers, for example, agreed never to release software that has not met certain testing standards, this would prevent market pressures from driving down the quality of software being produced. Davis's point is that the special responsibilities of professionals are grounded in what members of a professional group owe to one another; they owe it to one another to live up to agreed-upon rules and standards. Yet other theorists have tried to ground the special responsibilities of professionals in ordinary morality. Alpern [1991] argues, for example, that the engineer's responsibility for safety derives from the ordinary moral edict, *do no harm*. Since engineers are in a position to do greater harm than others, engineers have a special responsibility in their work to take greater care.

In the case of the role of computing professionals, responsibilities are not always well articulated because of several factors. Computing is a relatively new field. Moreover, many computer scientists and engineers are both employees of companies and simultaneously members of a profession. This can create a tension blurring a professional's responsibilities. Being a professional means having the independence to make decisions on the basis of special expertise, but being an employee of a company often means acting for the best interests of the company, being loyal, and so on. The demands of a business (expectations of one's employer) can conflict with the demands of professional responsibility.

Another difficulty in defining and maintaining professional ethics for computing professionals is the diversity of the field. Computing professionals are employed in a wide variety of contexts, have a wide variety of expertise, and come from diverse educational backgrounds. There is no single unifying organization, no uniform admission standards, and no single identifiable professional role. To be sure, there are signs of the field moving in the direction of more professionalization, but as yet, computing is still a loose cluster of overlapping fields composed of individuals following diverse educational and career paths, and engaged in a wide variety of job activities.

Despite the lack of well-articulated unifying professional standards and ideals, there are expectations for professional practice. It is these expectations that form the basis of an emerging professional ethic that may, in the future, be refined to the point where there will be a strongly differentiated role for computer professionals.

These expectations, in particular their evolving character, can be seen in the growing sophistication of ethical codes in the field of computing. Professional codes play an important role in articulating a collective sense of what is both the ideal of the profession as well as the minimum standards required. Codes of conduct state the consensus views of members as well as shaping behavior.

A number of professional organizations have codes of ethics that are of interest here. The most well known include the Association for Computing Machinery (ACM) Code of Ethics and Professional Conduct (see Appendix B), the Institute of Electrical and Electronic Engineers (IEEE) Code of Ethics, the Data Processing Managers Association (DPMA) Code of Ethics and Standards of Conduct, the Institute for Certification of Computer

Professionals (ICCP) Code of Ethics, the Canadian Information Processing Society Code of Ethics, and the British Computer Society Code of Conduct. Each of these codes has different emphases and goals. Each in its own way, however, deals with issues that arise in the context in which computer scientists and engineers typically practice.

The codes are relatively consistent in identifying computer professionals as having responsibilities to be faithful to their employers, to clients, and to protect public safety and welfare. The most salient ethical issues that arise in professional practice have to do with balancing these responsibilities together with personal (or nonprofessional) responsibilities. Two common areas of tension are worth mentioning here, albeit briefly.

As previously mentioned, computer engineers and scientists may find themselves in situations where their responsibility as professionals to protect the public comes in conflict with loyalty to their employer. Such situations sometimes escalate to the point where the computer professional has to decide whether to blow the whistle. Such a situation might arise, for example, when the computer professional believes that a piece of software has not been tested enough but her employer wants to deliver the software on time and within the allocated budget (which means immediate release and no more resources being spent on the project). The decision to blow the whistle or not to blow the whistle is one of the most difficult computer engineers and scientists may have to face. Whistle blowing has received a good deal of attention in the popular press and in the literature on professional ethics because this tension seems to be built into the role of engineers and scientists. Ideally, corporations and professional societies will, in the future, develop mechanisms to help avoid the need to blow the whistle. For example, if corporations had ombudspersons to whom engineers and scientists could report their concerns (anonymously) or if professional societies maintained hotlines that professionals could call for advice on how to get their concerns addressed, these would lessen the need to blow the whistle.

Another important professional ethics issue that often arises is directly tied to the importance of being worthy of client (and indirectly public) trust. Professionals can find themselves in situations in which they have (or are likely to have) a conflict of interest. A conflict of interest situation is one in which the professional is hired to perform work for a client and the professional has some personal or professional interest that may (or may appear to) interfere with their judgement on behalf of the client. For example, suppose a computer professional is hired by a company to evaluate their needs and recommend hardware and software that will best suit the company's needs. The computer professional does precisely what is requested, but fails to mention being a silent partner in a company that manufactures the hardware and software that has been recommended. In other words, the professional has a personal interest—financial benefit—in the company buying certain equipment. If the company were to find this out later on, it might rightly be thought that there had been deception. The professional was hired to evaluate the needs of the company and to determine how best to meet them, and in so doing to have the best interests of the company fully in mind. Now it is suspected that the professional's judgment may have been biased. The professional had an interest that might have biased his or her judgement.

There are a number of strategies that professions use to avoid these situations. A code of conduct may, for example, specify that professionals reveal all relevant interests to their clients before they accept a job. Or, the code might specify that members never work in a situation where there is even the appearance of a conflict of interest.

This brings us to the special character of computer technology and the effects that the work of computer professionals can have on the shape of our world. Some may argue that computer professionals have very little say in what technologies get designed and built. This seems to be mistaken on at least two counts. First, we can distinguish computer professionals as individuals and computer professionals as a group. Even if individuals have little power

in the jobs they hold, they can exert power collectively. Second, individuals can have an effect if they think of themselves as professionals and consider it their responsibility to think about the impact of their work.

2.4 Ethical Issues that Arise from Computer Technology

The effects of a new technology on society can draw attention to an old issue, and can change our understanding of that issue. The issues listed in this section—privacy, property rights, risk and reliability, and global communication—were of concern, even problematic, before computers were an important technology. But computing and, more generally, electronic telecommunications, have added new twists and new intensity to each of these issues. Although computer professionals cannot be expected to be experts on all of these impacts, it is important for them to understand how computer technology is shaping the world. And, it is important for them to keep these impacts in mind as they work with computer technology. Those who are aware of privacy issues are more likely to take this into account when they design database management systems, those who are aware of risk and reliability issues are more likely to articulate these to clients and attend to them in design and documentation, and so on.

2.4.1 Privacy

Privacy is a central topic in computer ethics. Some have even suggested that privacy is a notion that has been antiquated by technology and that it should be replaced by a new openness. Others think that computers must be harnessed to help restore as much privacy as possible to our society. Although they may not like it, computer professionals are at the center of this controversy. Some are designers of the systems that facilitate information gathering and manipulation; others maintain and protect the information. As the saying goes, *information is power*, but power can be used and/or abused.

Computer technology creates wide ranging possibilities for tracking and monitoring of human behavior. Consider just two ways in which personal privacy may be affected by computer technology. First, because of the capacity of computers, massive amounts of information can be gathered by record keeping organizations such as banks, insurance companies, government agencies, educational institutions. The information gathered can be kept and used indefinitely, and shared with other organizations, rapidly and frequently. A second way in which computers have enhanced the possibilities for monitoring and tracking of individuals is by making possible new kinds of information. When activities are done using a computer, transactional information is created. When individuals use automated bank teller machines, records are created; when certain software is operating, keystrokes on a computer keyboard are recorded; the content and destination of electronic mail can be tracked, and so on. With the assistance of newer technologies, much more of this transactional information is likely to be created. For example, television advertisers may be able to monitor television watchers with scanning devices that record who is sitting in a room facing the television, and new highway systems may allow drivers to pass through toll booths without stopping as a beam reading a bar code on the automobile will automatically charge the toll to a credit card, creating a record of individual travel patterns. All of this information (transactional and otherwise) can be brought together to create a detailed portrait of a person's life, a portrait the individual may never see, though it is used by others to make decisions about the individual.

This picture of computer technology suggests that computer technology poses a serious threat to personal privacy. However, one can counter this picture in a number of ways. Is

it computer technology per se that poses the threat or is it just the way the technology has been used (and is likely to be used in the future)? Only those who understand the technology are in a position to design or change computer technology so that it does not eradicate privacy.

At the same time we think about changing technology, we also have to ask deeper questions about privacy itself and what it is that individuals need, want, or are entitled to when they express concerns about the loss of privacy. In this sense, computers and privacy issues are ethical issues. They compel us to ask deep questions about what makes for a good and just society. Should individuals have more choice about who has what information about them? What is the proper relationship between citizens and government, between individuals and private corporations? As previously suggested, the questions are not completely new; but some of the possibilities created by computers are new, and these possibilities do not readily fit the concepts and frameworks used in the past.

2.4.2 Property Rights and Computing

The protection of intellectual property rights has become an active legal and ethical debate, involving national and international players. Should software be copyrighted, patented, or free? Is computer software a process, a creative work, a mathematical formalism, an idea, or some of all of these? What is society's stake in protecting software rights? What is society's stake in widely disseminating software? How do corporations and other institutions protect their rights to ideas developed by individuals, and what are the individuals' rights? These kinds of questions must be answered publicly through legislation, through corporate policies, and with the advice of computing professionals. Some of the answers will involve technical details, and all should be informed by ethical analysis and debate.

Perhaps the issue that has received the most legal and public attention is that concerning the ownership of software. In the course of history, software is a relatively new entity. Whereas western legal systems have developed property laws that encourage invention by granting certain rights to inventors, there are provisions against ownership of things that might interfere with the development of the technological arts and sciences. For this reason, copyrights protect only the expression of ideas, not the ideas themselves, and we do not grant patents on laws of nature, mathematical formulas, and abstract ideas. The problem with computer software is that it has not been clear that we could grant ownership of it without, in effect, granting ownership of numerical sequences or mental steps. Software can be copyrighted, because a copyright gives the holder ownership of the *expression* of the idea (not the idea itself), but this does not give software inventors as much protection as they claim to need to *fairly* compete. Competitors may see the software, grasp the idea, and write a somewhat different program to do the same thing. The competitor can sell the software at less cost because the cost of developing the first software does not have to be paid. Patenting would provide stronger protection, but until quite recently the courts have been reluctant to grant this protection because of the problem previously mentioned: patents on software would appear to give the holder control of the building blocks of the technology, an ownership comparable to owning ideas themselves.

Like the questions surrounding privacy, property rights in computer software also lead back to broader ethical and philosophical questions about what constitutes a just society. In computing, as in other areas of technology, we want a system of property rights that promotes invention (creativity, progress), but at the same time, we want a system that is fair in the sense that it rewards those who make significant contributions but does not give anyone so much control that they prevent others from creating. Policies with regard to property rights in computer software cannot be made without an understanding of the technology, and this is why it is so important for computer professionals to be involved in

public discussion and policy setting on this topic.

2.4.3 Risk, Reliability, and Accountability

As computer technology becomes more important to the way we live, its risks become more worrisome. System errors can lead to physical danger, sometimes catastrophic in scale. There are security risks due to hackers and crackers. Unreliable data as well as intentional misinformation are risks that are increased because of the technical and economic characteristics of digital data. Furthermore, the use of computer programs is, in a practical sense, inherently unreliable.

Each of these issues (and many more) requires computer professionals to face the linked problems of risk, reliability, and accountability. Professionals must be candid about the risks of a particular application or system. Computing professionals should take the lead in educating customers and the public about what predictions we can and cannot make about software and hardware reliability. Computer professionals should make realistic assessments about costs and benefits, and be willing to take on both for projects they are involved with.

There are also issues of sharing risks as well as resources. Should liability fall to the individual who buys software or to the corporation that developed it? Should society acknowledge the inherent risks in using software in life-critical situations and shoulder some of the responsibility when something goes wrong? Or should software providers (both individuals and institutions) be exclusively responsible for software safety? All of these issues require us to look at the interaction of technical decisions, human consequences, rights, and responsibilities. They call not just for technical solutions but for solutions which recognize the kind of society we want to have and the values we want to preserve.

2.4.4 Rapidly Evolving Globally Networked Telecommunications

The system of computers and connections known as the Internet is forming a new kind of community or sets of communities—electronic communities. Questions of individual accountability and social control, as well as matters of etiquette that arise in all [**?**] societies are taking shape in a new way, in the electronic medium. It is as if we have society (societies) forming in a new physical environment. A new way of living together is evolving as we watch. What will the Internet be like in five years? Who will and won't have access? How much freedom will we trade for security? How will commercial interests and citizens opposed to commercialization coexist? What will electronic communications mean to our worlds of work and play? Will the Internet begin to change who we are or who we are to each other?

Speculating about the Internet is now a popular pastime. But some researchers in computer ethics think that more serious thought, and perhaps action, should be applied to shaping the society of network users. Commercial, governmental, and recreational groups are already changing what the Internet used to be, often making unilateral statements or actions. Instead of asking What will happen to the Internet? we should perhaps be asking What *should* happen to the Internet? Questions of *should* are exactly the questions that ethics addresses.

2.5 Final Thoughts

Computer technology will, no doubt, continue to evolve and will continue to affect the character of the world we live in. Computer scientists and engineers will play an important role in shaping the technology. The technologies we use shape how we live and who we are.

They make every difference in the moral environment in which we live. Hence, it seems of utmost importance that computer scientists and engineers understand just how their work affects humans and human values.