Editorial: Strategic Directions in Computing Research

PETER WEGNER

Brown University ⟨*pw@cs.brown.edu*⟩

JON DOYLE

Massachusetts Institute of Technology ⟨*doyle@mit.edu*⟩

Computing Surveys is commemorating the 50th anniversary of the ACM and of the computing discipline in two special issues. The March 1996 issue, on "Perspectives in Computer Science," examined the status of the discipline, while the present issue looks to the future with a collection of reports on "Strategic Directions in Computing Research." These reports evolved from a workshop, hosted by the MIT Laboratory for Computer Science in June 1996, at which 22 working groups consisting of more than 300 participants met to examine research directions. The preparation of reports in the subsequent months proved more time-consuming than expected, but 19 of the working groups completed their reports for publication in this issue. The reports collectively provide a remarkably deep, though incomplete, view of the field and its future challenges. We hope they will stimulate further efforts toward strategic understanding, and we offer the pages of *Computing Surveys* as a home for future strategic-directions reports. The September or December 1997 *Surveys* will contain a symposium on strategic directions in computing research that reacts to the articles in this issue. Readers interested in submitting short (1000-word) articles to such a symposium should so inform the editors ⟨csur@acm.org or pw@cs.brown.edu⟩.

Over the past 30 years computers have become an increasingly important part of everyday life. The success of computer technology has inevitably altered the role of core research. The ACM has changed from a small society representing researchers to a large professional organization in which researchers comprise less than 20% of its members. The *Communications of the ACM* has changed its format from a scholarly journal to a magazine, while *Computing Surveys*, though preserving its scholarly character, is also changing to better serve the changed membership profile. The evolving role of computing in society affects the self-image of researchers as well as the research philosophies of funding agencies and governments.

The report *Computing the Future* [Hartmanis and Lin 1992], which recognized the changing role of research in the discipline of computing, recommended balancing its first priority—sustaining core research—with an effort by researchers to broaden the field, so as to play a greater role in an expanding discipline and to enrich computing models through contact with applications. These recommendations engendered some controversy because they struck some researchers as favoring short-term over long-term research. Such fears of overemphasis on short-term results at the expense of long-term research are legitimate and need to be addressed. The reports in this issue balance the desire of researchers to undertake core research with the need to build bridges connecting theory and practice. The tension between supply-driven core research—focused on increasing understanding and exploring new possibilities—and demand-driven applications research—focused on solving external problems—is a permanent part of the strategic landscape. We hope that these reports help the computing community better to appreciate the scope and importance of research.

The 19 reports are grouped into three broad, though overlapping, areas: foundations, systems, and applications and infrastructure:

Foundations: six reports on theory of computation; computational geometry; concurrency; formal methods; programming languages; artificial intelligence.

Systems: nine reports on computer architecture; networks and telecommunications; object-oriented programming; constraint programming; software engineering and programming languages; software quality; real-time and embedded systems; database systems; storage I/O issues in large-scale computation.

Applications and Infrastructure: four reports on human-computer interaction; computational science and engineering; electronic commerce and digital libraries; computer science education.

Though the boundaries between these categories are imprecise, this classification is nevertheless useful in expressing that computer science builds systems—anchored in foundations—that provide an infrastructure for domain-specific applications. The area of foundations combines the traditional areas of algorithms and complexity with the broader foundational areas of formal methods, programming languages, and artificial intelligence. Topics under systems focus on the organization and implementation of services of computing systems. Topics in the third category address the technical and social infrastructure of computing via human-computer interaction and computer science education, and treat applications crucial in the technical and commercial infrastructure of society via computational science and engineering, and electronic commerce and digital libraries. Each area needs the others to make it meaningful, and each report in fact strikes a balance among foundations, systems, and applications and infrastructure.

FOUNDATIONS

The area of foundations is concerned with the broad range of fundamental concepts and principles that provide a rigorous and systematic framework for the study of systems and applications. Many strategic areas of computing research support and contribute to foundations in this broad sense, even when the research goals are primarily practical.

The *theory of computing* report focuses on core research in algorithms and complexity and examines the considerable accomplishments of algorithms in applications like security, information retrieval, communication networks, statistical physics, molecular biology, and biochemistry. The report examines achievable performance of algorithms (upper bounds), inherent difficulty (lower bounds), reducibility among algorithms (NP-complete and other complexity classes), and major open questions (is P = NP?). Although algorithms dominated the theoretical perspective of theorists like Turing, Knuth, and workers in combinatorial algorithms and complexity theory, and although algorithms will always be a central foundation of computing, theoretical computing research consists of an evolving body of principles broader than algorithms, and has spawned conferences in areas such as concurrency and logic in computing. The five other reports classified under foundations give the flavor of additional theoretical areas, though they do not completely cover areas like logic and semantics.

Computational geometry, which investigates algorithms for geometric problems, clearly demonstrates the impact of theory on practice in a particular domain. Computational geometry has proved both a fertile application area and a driver of theoretical advances in algorithmic analysis. The report highlights impressive accomplishments in 14 different problem areas, including motion planning and parallel algorithms, during this subfield's short 20-year existence. It addresses the methodology of robustness, fine-grain complexity analysis, implementation, and experimentation, discusses paradigms of distributed, real-time, and randomized computing, and examines visualization, graph drawing, animation, and geographic information systems. Computational geometry emerges as a cohesive application area of algorithmic analysis that illustrates how intelligent application of algorithmic techniques can provide a systematic approach to the formulation and efficient solution of large classes of new and interesting problems.

Concurrency is an entirely different subarea of computing, concerned with largely nonalgorithmic models and logics of parallel computation. It treats problems arising from multiple threads of control and from ongoing interaction with an environment. The report focuses on concurrency theory and its use in verification, rather than on applications to concurrency control, operating systems, debugging, distributed systems, or real-time computing. In exploring process models it contrasts intensional (state-transition) and extensional (observation-based) models, interleaving and true concurrency, and branching and linear time. Interleaving models reduce concurrency to equivalent sequential algorithmic processes, while true concurrency models treat concurrency as a primitive irreducible notion. Temporal logics and verification algorithms for safety and liveness are classified according to whether they use linear or branching time. The report examines alternative unifying semantic frameworks for concurrency and briefly explores strategic directions for concurrent programming languages, design and verification methodologies, and education.

Formal methods for the specification and verification of hardware and software systems are becoming increasingly important as systems increase in size and complexity. The report illustrates progress in formal methods through an impressive collection of industrial-strength examples that contrast sharply with the toy examples of early work in the area. Progress has resulted from improved specification formalisms, model-checking techniques for verification of finite models, and improved theorem-proving techniques. The report briefly describes these techniques and their tool support. It identifies the need for further conceptual work on fundamental concepts like composition, abstraction, and reusable models and theories; the need for tools that are incremental, efficient, usable, integrable, and have a common look and feel; and the need to integrate formal methods more seamlessly into the entire system development cycle. In filling these needs, there is great promise of further practical benefits.

The *programming language* report examines strategic directions in semantics, types, static program analysis, program transformation, and implementation. It focuses on foundations, leaving to other reports areas like object-oriented and constraint programming and software engineering. Programming languages are a primary tool for thinking about problem formulation and programming. The area of programming languages has always been difficult to classify because it deals both with issues of semantics and types that are central areas of the theory of computation and with questions of design, implementation, and usability that are eminently practical. The classification of programming languages by paradigm into imperative, functional, logic, constraint, object-oriented, and concurrent language classes, which nicely captures major differences of style in problem-solving and programming, is not suitable for classifying strategic research directions since the research categories considered cut across paradigms. Users increasingly interact with computers through graphical user interfaces rather than programming languages, and this may cause a change in emphasis in the languages and semantic frameworks to be studied.

Artificial intelligence is concerned with foundational issues of formalizing knowledge and its use in building intelligent systems and machines, as well as the scientific and practical aims of constructing computational models of human behavior and creating an infrastructure that makes computers easier to use. It has succeeded to varying degrees in building systems that solve intellectual problems, control robot motions, understand natural language, and learn from experience. The report divides the field into ten primary areas, including knowledge representation, learning, planning, language and image understanding, robotic motor control, autonomous agents, cognitive modeling, and mathematical foundations. It describes several major directions, influencing research in all these areas rather than summarizing each individually. These major directions include building robots, modeling rationality, supporting collaboration, communication, and knowledge acquisition, and integrating techniques drawn from different subareas, from other areas of computing research (such as databases) and from other disciplines (such as economics). The beginnings of artificial intelligence antedate computer science, going back to automata like the Golem and questions like "Can machines think?" Its impact on mainstream computer science has been considerable through languages like Lisp, which was developed in the 1950s at the same time as Fortran. Though much of the practical impact of artificial intelligence has been slower to develop and is less visible than in other areas of computing, advancing computing power and progress of the field now support an increasingly wide range of successful industrial-strength applications and technologies.

SYSTEMS

Computing systems were the primary subject of early computer science, since building effective compilers and operating systems was a primary bottleneck to the expanded use of computers. The scope of the system area has steadily expanded to include databases, networks, and software engineering. Systems extend the functionality of hardware by imposing a logical structure on resources delivered to users which builds on the physical structure of underlying computers. They provide a common substrate of services for application programming. Though research in systems initially focused on the building and maintenance of usable software, theory plays an increasing role in developing principles and models, while new kinds of applications made possible by advances in technology drive the development of new kinds of systems.

Computer architecture is concerned with the structure and design principles of major components of a computing system. Advances in hardware architecture are driven by the technology of semiconductors, VLSI, and telecommunication, and by applications like the World-Wide Web, distributed systems, multimedia, and virtual reality. Architectural challenges in reducing power consumption and complexity, while increasing performance, complement software challenges. Architectural technologies like pipelining, superscalar, and very long instruction words, first proposed for supercomputers, are being applied to microprocessors as they become more powerful. Massively parallel and high-performance computing, which received strong support under the U.S. high-performance computing initiative, has become less fashionable, in part because of commercial failures. Parallel processing has succeeded at the instruction and networking level with Pentium, Alpha, and MIPS-R10000 technologies, rather than at the supercomputer level. Memory hierarchy design is an important area addressed in another report. Moore's law of hardware improvement by a factor of 2 every 18 months shows no sign of abating, though performance improvements will eventually run into physical barriers due to the speed of light. Though architecture will continue to be a challenging and strategically important research area, software will increasingly be the bottleneck in computer application development.

The area of *networks and telecommunications* spans both architecture and software. Networking technologies initially developed for telecommunications are being adapted to computer communication: circuit technology has been replaced by packet technology with connection-oriented packets, whose switching is accomplished by the network, or with datagrams, whose switching information is contained in the packet itself. Questions of architecture, protocols, reliability, and scalability have different solutions for information networks and must be further developed to accommodate multimedia and the low latency required for virtual reality. Networking models collaborative processes closely tied to emerging research areas of coordination and interoperability. Questions of congestion control, routing, and signaling protocols, which were central in maintaining efficient telephone networks, are still important, but many new issues must be considered in areas like network security, mobile processes and wireless communication, and security protection, both against viruses and to facilitate electronic commerce.

Object-oriented programming is viewed as a modeling technique that better captures the correspondence with objects of application domains than procedure-oriented programming. It promotes reusability, extensibility, and adaptability, and is robust in uniformly modeling the lifecycle phases of specification, design, implementation, and enhancement in an integrated way. The report examines work on foundations, languages, environments, tools, architecture, and design. It identifies technology integration, software components, patterns and frameworks, and distributed programming as important research areas. Future directions include adaptive programming, object composition, reflection, and aspect-oriented programming (which captures the notion that modeling and implementation have multiple aspects, facets, and interfaces). Though object-oriented programming in the narrow sense is classified as a subfield of programming languages, it is increasingly viewed by its practitioners as a paradigm for modeling and software engineering that more directly models domain-specific applications than imperative structured programming.

Constraint programming is a paradigm built around the mathematical concept of relational constraints. Though it evolved as a generalization of logic programming, its basic concept—problem solving by progressively constraining a space of possibilities—is a general top-down paradigm that may be contrasted to the bottom-up paradigm of developing an algorithm for a specific solution. The report combines the presentation of fundamental concepts with an in-depth discussion of applications to artificial intelligence, databases, user interfaces, operations research, concurrency, robotics, and control theory. It classifies constraints by whether they are Boolean, finite, real-valued, linear, global, or user-defined. The report also examines constraint-based tools for programming, debugging, and visualization and discusses implementation techniques for constraint-based programming languages. This report, like many of the others, provides a synthesis both of the current state of the art and of the strategic directions along which the field will progress over the next few years.

The report on *software engineering and programming languages* examines the role of programming languages in controlling software complexity and lifecycle cost, paying particular attention to synergy between the software-engineering and programming-language communities in the areas of programming the Web, domain-specific programming, specification, and program analysis. Web issues include security, distributed software development, mobility, and global requirements. Domain-specific issues include formalizing informal notions, integrating multiple specifications, domain specialization of best practices, and optimization. Specification research includes methods of precise description, formal methods, partial specification of key properties, and development of specification languages. Program analysis includes a wide range of issues relating to reengineering, binding time, representation management, and integrity checking.

The goal of affordable development of safe, dependable, and usable software has proved elusive, in part because *software quality* has dimensions of robustness, usability, and efficiency that transcend correctness and are difficult to specify and verify. Formal methods for correctness specification and verification have made progress, but formalization is difficult even for correctness and is even harder for other software qualities. The report examines the state of the art in dynamic testing, static analysis, symbolic execution, formal methods, experimental evaluation and measurement, and a number of other areas. Software quality practice relies on empirical techniques like regression testing. An important strategic direction in achieving software quality will be research on built-in quality starting at the early lifecycle phases. Postdevelopment testing and analysis has serious limitations that are well recognized, and should be supplemented by more analysis and testing during design. Movement of research developments into practice has been slow, and progress in achieving better software quality will require greater synergy between the research community and practitioners than has been achieved in the past.

Real-time computing is an enabling technology for process control, manufacturing, avionics, multimedia, virtual reality, defense applications, and many other areas. Real-time systems requirements transcend functional correctness and include safety-critical soft and hard real-time requirements. The report presents some illuminating case studies and examines issues of systems evolution, open real-time systems, composability, software engineering, timeliness and reliability guarantees, formal verification, multimedia, programming languages, and education. In the near future almost all products and engineering processes will contain real-time features and embedded processes, and the demand for safe, certifiable, and dependable real-time systems with high quality of service will be overwhelming.

The topic of *databases* is concerned with building database systems and also with data management, which is a core application technology for managing the information infrastructure. The forms of data storage and management are undergoing radical changes as a consequence of changes in information technology. Data management is becoming increasingly independent of databases: for example, the data management problems of the World-Wide Web are rarely handled by classical database management systems. The report concludes that data management should be unbundled from database management and that the incremental tools for managing data in distributed networked environments may have very different requirements than classical database-management tools. The report presents case studies of virtual enterprises and personal information systems to illustrate the need for unbundled tools. It examines questions of scale, scheme organization, data quality, heterogeneity, query complexity, ease of use, and security from this point of view, as well as research topics that must be addressed to facilitate the technology transition to "lightweight" data management.

Storage I/O in large-scale computing is a new area whose identity as a separate field of study is defined for the first time in this report. The report makes a strong case that the management of large-scale storage systems determines an emerging strategic subfield of great technological importance. The body of principles and algorithms for storage I/O determines a domain-specific area of computing systems with close connections to several other areas, such as databases, distributed systems, architecture, real-time systems, algorithms, parallel computing, compilers, operating systems, and file systems, for which I/O performance is a prime scalability factor. Issues of performance, persistence, reliability, scale, and usability are examined for cached, historical, multimedia, and scientific data. The report puts forward as a strategic goal a unified body of storage access and management techniques in many specific application areas. Technical research towards this goal includes work on virtual device drivers, new program abstractions, exploiting data types and access paths, prefetching, caching, and scheduling, performance and behavior models, and a variety of other techniques.

APPLICATIONS AND INFRASTRUCTURE

Applications and infrastructure motivate research in foundations and systems and provide a measure of success. Applications provide problems and case studies that suggest new kinds of theoretical research and allow such research to be grounded in reality. Human-computer interaction deals with domain-independent principles and infrastructure for interacting with computers. Computational science and engineering was one of the earliest application areas, which today provides the key infrastructure of science and technology as well as being a catalyst for high-performance computing. Electronic commerce and digital libraries provide infrastructure for broad domain-specific application areas. Education is concerned with maintaining and improving the infrastructure of human and social knowledge and nurturing human skills that determines progress in all areas of computing.

Human-computer interaction (HCI) provides domain-independent infrastructure to facilitate effective communication between people and computers. It addresses ease of use and interface technology, as well as broader questions of how computers affect individuals, organizations, and society. The report reviews the evolution of windows, hypertext, user-interface, and toolkit technologies and the connection of HCI to psychology, linguistics, artificial intelligence, and anthropology. It examines the relation of HCI to progress in database technology, education, and lifelong learning, electronic commerce, end-user programming, information visualization, and computer-mediated communication. Technological trends that impact HCI include ubiquitous computing, speed, size, and bandwidth, input modalities like speech, handwriting, natural language, and three-dimensional environments and virtual reality. The look-and-feel of the future information infrastructure and of an information marketplace for commerce, entertainment, and education will depend heavily on HCI research and technology.

Computational science and engineering, a dominant early application area of computing is experiencing a resurgence as computing power expands to make possible the solution of increasingly complex scientific and engineering problems. It played an important part in motivating the U.S. high-performance computing initiative through grand challenges in weather, materials sciences, human genome, astronomy, and transportation problems. The report presents case studies relating to atomic structure of viruses and proteins, modeling biomolecules, structural engineering, computational electromagnetics, computational medicine, fluid flow, and computational finance. Computational science and engineering can reduce duplication of algorithms across applications and build on advances in sequential and parallel computation to solve increasingly large and important computational problems. Parallel computing has not realized its promise, however, and has in fact suffered a commercial slump, in part because of a lack of coordination between computer and computational science. High-performance computing addressed the problem of scalability for existing problems but was not as successful in expanding the scope and complexity of problems that could be solved. The impedance mismatch between large-scale and large-scope computing cannot be resolved without greater attention by computer scientists to the domain-specific problems of computational scientists. Challenging large-scope problems include microscale electromechanical systems, large mechanical systems like airplanes, automobiles, and robots, natural environments and data mining. Enabling technologies that computer scientists can provide for computational science include interactive compilers, parallel programming languages, graphics and visualization tools, performance evaluation tools, distributed operating systems, innovative architectures, and database management. The report also criticizes the publication style for algorithms in computer science as not being in a form that computational scientists can easily apply to their problems. This article deserves careful reading by computer scientists for both its constructive suggestions and its criticisms of how the computer science community interacts with application programmers.

Electronic commerce and digital libraries are two major components of the emerging global open marketplace (digital agora) for information, goods, and services. The report examines both common features of electronic commerce and digital libraries that provide the infrastructure of the information marketplace and domain-specific features of electronic commerce and digital libraries. The infrastructure challenges include tools for acquiring, storing, and filtering information, feature extraction, quality of information, agents and matchmaking services, domain-specific ontologies, data mining, and query management. The report also considers security and confidentiality, particularly important for electronic commerce, and discusses the importance of new forms of cost management and laws concerning intellectual property. It also discusses the socioeconomic and legal impacts of relying on cyberspace rather than paper for commerce and other everyday needs. Three case studies of electronic commerce and six digital library projects are presented. The domain-specific discussion of this report nicely complements the more abstract discussion of infrastructure requirements in the HCI and database reports.

Education in computer science and engineering will determine the priorities and competence of the next generation of computing professionals. The report reviews curriculum development from ACM's "Curriculum 68" to the ACM/IEEE-sponsored "Computing Curricula 1991," discusses general curriculum issues such as adaptation in a rapidly changing field and sharing of educational resources, and examines K-12, undergraduate, and graduate issues and coordination within the education community. The report addresses the balance between research and teaching and between long-term and industrial needs, and recommends more emphasis on the needs of industry in advanced courses. Graduate education, especially at the master's level, should combine teaching of basic principles with training for the kinds of jobs that students are likely to take when they finish. The report recommends more interaction among organizations that represent education, and makes a specific proposal for a center for computer science and engineering education that facilitates the sharing of computational resources. There is much ferment and some confusion at all levels concerning the teaching of computing, with no agreement on how it should be taught in high school, at the advanced placement level, in first courses, or in junior-senior courses. The number of undergraduate students in U.S. institutions reached a peak around 1986, experienced a dip in the late 1980s and early 1990s, but now seems to be equaling or exceeding the mid-1980s levels. The need for more uniform standards and a better understanding of what we should teach is certainly great, and new initiatives are needed to develop relevant curricula that meet the needs of the 21st century.

PERSONAL POSITION STATEMENTS

Most participants at the Strategic Directions in Computing Research workshop wrote personal statements before the meeting to help shape the discussions at the meeting. The chairs of each working group have acted as editors in reviewing and selecting many of these statements for electronic publication in the online volume described below.

NEW ONLINE SECTIONS

Starting with this issue, *Computing Surveys* inaugurates a new series of online sections of issues to complement the printed volumes. The first such section, Volume 28(4es), contains personal statements from some of the participants at the Strategic Directions in Computing Research workshop. The chairs of each working group acted as editors in selecting and reviewing these statements for publication.

Readers may obtain the online sections through the *Computing Surveys* web page located at http://www.acm.org/surveys, together with online versions of some of the printed *Surveys* articles. The editors intend that articles published in the online sections share the same quality and reviewing standards as articles published in the printed issues, and may be cited in comparable ways (see, for example, the ACM guidelines for citation of online articles, published at http://www.acm.org/pubs/citations.html). The table of contents of this issue lists the articles appearing in Volume 28, Number (4es) (December 1996).

ACKNOWLEDGMENTS

The Strategic Directions workshop, from which these reports derive, received major financial support from ACM, the Computing Research Association, the National Science Foundation, and the Office of Naval Research, as well as important logistical help from Brown University and the MIT Laboratory for Computer Science. We especially thank Katrina Avery for her editorial help, John Bazik and Scott Lewandowski for their technical help, and Michael Dertouzous, Bob Donahue, Dina Goldin, Heather Grove, Jennifer Kratochwill, Phillip Le, Michael Littman, Lissa Natkin, and Anne Wailes for their help in making the meeting a success.

The reports published here reflect much hard work by the chairs of the Strategic Directions working groups, who selected their group members and organized the writing (and rewriting) of their reports, as well as reviewing and acting as subeditors for the personal statements published in the online section. We thank them deeply for their efforts. We are also grateful to those who helped in reviewing and commenting on group reports. We thank Chris Hankin, Peter Denning, and Larry Rudolph for their work here. Finally, we thank the authors of position statements, both for their help in formulating the group reports and for their

patience in preparing the contributions to the new online section of volume 28, Number 4.

REFERENCES

HARTMANIS, J. AND LIN, H. EDITORS. *Computing the Future: A Broader Agenda for Computer Science and Engineering*. National Academy Press, 1992.