Computer Engineering Curriculum in the New Millennium

Andrew McGettrick, Mitchell D. Theys, *Member, IEEE*, David L. Soldan, *Fellow, IEEE*, and Pradip K. Srimani, *Fellow, IEEE*

***Abstract*—Currently there is a joint activity (referred to as Computing Curricula 2001, shortened to CC2001) involving the Association for Computing Machinery and the IEEE Computer Society, which is producing curriculum guidance for the broad area of computing. Within this activity, a volume on Computer Engineering is being developed. This volume addresses the important area of the design and development of computers and computer-based systems. Current curricula must be capable of evolving to meet the more immediate needs of students and industry. The purpose of this paper is to look at areas of future development in computer engineering in the next ten years (2013) and beyond and to consider the work of the Computer Engineering volume of CC2001 in this context.**

***Index Terms*—Computing Curricula 2001 (CC2001), computer engineering curriculum, IEEE Computer Society (IEEE-CS) and Association for Computing Machinery (ACM), vision for next decade.**

I. History

IN 1998, the Association for Computing Machinery (ACM) and the Computer Society of the Institute for Electrical and Electronics Engineers (IEEE-CS) convened a joint curriculum task force called *Computing Curricula 2001*, or *CC2001* for short. The CC2001 Task Force was asked to develop a set of curricular guidelines that would match the latest developments of computing technologies in the past decade and endure through the next decade.

Over the past 50 years, *computing* has become an extremely broad designation that extends well beyond the boundaries of computer science to encompass many independent disciplines, including computer engineering, software engineering, information systems, and many others. The final report is planned to be organized into five volumes: Overview, Computer Science, Computer Engineering, Software Engineering, and Information Systems. Of these, the volumes on Computer Science and Information Systems have already been published. The others are in the process of being developed.

A committee was established in the beginning of 2001 to define the body of knowledge that constitutes computer engineering as well as to flesh out course outlines to suit different curricula in computer engineering. The purpose of this paper is to delineate the mission and the vision of the task and to invite members in particular and public in general to participate in this important activity.

II. Introductory Comments

Computer engineering embodies the science and the technology of the design, construction, implementation, and maintenance of the hardware and the software components of modern computing systems and computer-controlled equipment. Computer engineers are solidly grounded in the theories and principles of computing, mathematics, and engineering; they apply these theoretical principles to design hardware, software, networks, and computerized equipment and instruments to solve technical problems in diverse application domains. Continuing dramatic advances in computing and digital systems design have created opportunities for computer engineering professionals to apply those developments to the entire range of applications in engineering.

Over the past three decades, the discipline of computer engineering has emerged from the erstwhile fields of electrical engineering and computer science as a separate, although intimately related, discipline. The relevant professional societies, such as the IEEE-CS and the ACM, must upgrade the curricular guidelines appropriately at the proper time.

Efforts to design model curricula for the computing discipline began in the 1960s. The first report came out from the ACM in 1968 [3], making detailed curricular recommendations for academic programs in computer science. IEEE-CS published a *Model Curriculum for Computer Science and Engineering* [5] in 1977. In fact, the curriculum efforts on the engineering side of computing started with the Computer Science in Electrical Engineering (COSINE) Committee [1]. The ACM revised its curriculum in 1978 [6], while the IEEE-CS did an update of its computer science and engineering curricula in 1983 [7]. It was the IEEE-CS Model Curriculum that "bridged the gap between software oriented and hardware oriented programs for the first time." The 1983 IEEE-CS Curriculum Report emphasized the laboratory requirements and Accreditation Board for Engineering & Technology (ABET) guidelines in both the computer science and the computer engineering profession. There have been other attempts to look at computer engineering curricula over time [9], [14], [16], [20]. During the late 1980s, the IEEE-CS and the ACM joined forces to undertake a more ambitious curriculum review that was eventually published as Curriculum'91 or CC-91 [10]. The CC-91 report used the word "Computing" in its title to reflect that the discipline has both

Manuscript received October 29, 2002; revised July 21, 2003.

A. McGettrick is with the Department of Computer and Information Science, University of Strathclyde, England, U.K.

M. D. Theys is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60680 USA.

D. L. Soldan is with the Department of Electrical and Computer Engineering, Kansas State University, Manhattan, KS 66506 USA.

P. K. Srimani is with the Department of Computer Science, Clemson University, Clemson, SC 29631 USA (e-mail: srimani@cs.clemson.edu).

Digital Object Identifier 10.1109/TE.2003.818755

an "engineering" and a "science" component. In Fall 1998, the IEEE-CS and the ACM appointed a joint task force on "Year 2001 Model Curricula for Computing: CC-2001" to

> "develop a revised and enhanced curriculum (CC-2001) that will match the latest developments of Computing Technologies in the past decade and sustain through the next decade."

Subsequently, the final report would be organized into five volumes: Overview, Computer Science, Computer Engineering, Software Engineering, and Information Systems. A committee was established at the beginning of 2001 *to define the body of knowledge that constitutes computer engineering as well as to flesh out course outlines to suit different curricula in computer engineering*.

The purpose of this paper is to look at areas of future development in computer engineering in the next ten years (to 2013) and beyond and to consider the work of the Computer Engineering volume of CC2001 in this context.

III. Vision and Mission

The Computer Engineering Task Force did not start from scratch. Instead, it planned to build on the work of its predecessors. There are many aspects of the older reports that the authors intend to retain as they develop the new curriculum.

- The articulation of individual knowledge units serves a valuable purpose in providing a framework for the design of individual courses and the curriculum as a whole.
- The integration of professional practice and design into the undergraduate curriculum along the lines outlined in the appendexes to Curriculum'91 is supported.

However, as the computing technologies have been changing faster than the ability to keep pace, and computer applications have changed the structure of society significantly, much thought must be invested in deciding and defining what will go into the recommended curriculum to produce computer engineers competent to further the technology and its applications to benefit mankind in the future. As curriculum developers, decisions must be made concerning which basic knowledge is essential to the performance of computer engineers at a required level of competence and, thus, must be retained. On the other hand, as a discipline grows more and more mature and the body of knowledge compounds daily, the curriculum cannot contain everything; future computer engineers must be equipped with essential knowledge and well-tested methods and techniques, not just transient technologies.

The Computer Engineering Task Force has adopted many of the principles from CC2001: Computer Science [11]. The following is a complete list.

1) *Computing is a broad field that extends well beyond the boundaries of computer engineering.* A single report that covers only computer engineering cannot address the full range of issues that colleges and universities must consider as they seek to address their computing curricula. Additional reports in this series will be required to cover other computing disciplines.
2) *Computer engineering draws its foundations from a wide variety of disciplines.* The undergraduate study of computer engineering requires students to utilize concepts from many different fields. All computer engineering students must learn to integrate theory and practice, to recognize the importance of abstraction, and to appreciate the value of good engineering design.
3) *The rapid evolution of computer engineering requires an ongoing review of the corresponding curriculum.* Given the pace of change in our discipline, the process of updating the curriculum once a decade has become unworkable. The professional associations in this discipline must establish an ongoing review process that allows individual components of the curriculum recommendations to be updated on a recurring basis.
4) *Development of a computer engineering curriculum must be sensitive to changes in technology, new developments in pedagogy, and the importance of lifelong learning.* In a field that evolves as rapidly as computer engineering, educational institutions must adopt explicit strategies for responding to change.
5) *The computer engineering curriculum must go beyond knowledge units to offer significant guidance in terms of individual course design.* It will be effective only to the extent that it defines a small set of alternative models—preferably between two and four—that assemble the knowledge units into reasonable, easily implemented courses. Articulating a set of well-defined models will make it easier for institutions to share pedagogical strategies and tools. It will also provide a framework for publishers who provide the textbooks and other materials for those courses.
6) *The computer engineering curriculum should seek to identify the fundamental skills and knowledge that all computer engineering students must possess.* Despite the enormous breadth of computer engineering, there are concepts and skills that are common to computer engineering as a whole. The computer engineering curriculum must attempt to define the common themes of the discipline and ascertain that all undergraduate programs include this material.
7) *The required body of knowledge must be made as small as possible.* As computer engineering has grown, the number of topics required in the undergraduate curriculum has grown as well. Over the last decade, computer engineering has expanded to such an extent that it is no longer possible simply to add new topics without taking others away. The best strategic approach is to *reduce* the number of topics in the required core so that it consists only of those topics for which there is a broad consensus that the topic is essential to undergraduate degrees. Coverage of the core is not limited to introductory courses but will extend throughout the curriculum. At the same time, it is important to recognize that this core does not constitute a complete undergraduate curriculum but must be supplemented by additional courses that may vary by institution, degree program, or individual student.

8) *The computer engineering curriculum must strive to be international in scope.* Although curricular requirements differ from country to country, the curriculum is intended to be useful to computing educators throughout the world. Although it will be strongly influenced by educational practice in the U.S., every effort will be made to ensure that the curriculum recommendations are sensitive to national and cultural differences so that they will be widely applicable throughout the world.
9) *The development of computer engineering curriculum must be broadly based.* To be successful, the process of creating the curriculum recommendations must include participation from many different constituencies, including industry, government, and the full range of higher educational institutions involved in computer engineering education.
10) *The computer engineering curriculum must include professional practice as an integral component of the undergraduate curriculum.* These practices encompass a wide range of activites, including management, ethics and values, written and oral communication, working as part of a team, and remaining current in a rapidly changing discipline.
11) *The computer engineering curriculum must include discussions of strategies and tactics for implementation along with high-level recommendations.* Although it is important for the *computer engineering curriculum* to articulate a broad vision of computing education, the success of any curriculum depends heavily on implementation details.
12) *The computer engineering curriculum core must acknowledge that engineering curricula should be accredited.* As such, the document must detail the core that all programs should have. The computer engineering curriculum should not just fit within ABET criteria, but instead should provide information to ABET about what the next iteration criteria should be.
13) *The computer engineering curriculum must include an appropriate and necessary design and laboratory experience component.* A computer engineer requires a laboratory experience that should provide problem-solving and debugging experience. In addition, it should provide information about alternative laboratory experiences for alternative students who are not on campus, such as distance learning and Internet courses.

IV. Knowledge Areas

The following content areas or knowledge areas have been identified along with a tentative (incomplete and preliminary) list of knowledge units. They are meant to be preliminary for the purpose of bootstrapping the discussions of focus groups formed to take on each knowledge area. The core knowledge units currently include the following:

- *SPR* (Social and Professional Issues): intellectual property, privacy and civil liberties, and economic issues in computing;
- *CAO* (Computer Architecture and Organization): computer arithmetic, memory system, organization and architecture, interfacing and communication, processor systems design, organization of the CPU, performance, and multiprocessing;
- *CSE* (Computer Systems Engineering): overview, theoretical considerations, life cycle, requirements analysis, architectural design, implementation, testing, maintenance, and hardware and software co-design;
- *SWE* (Software Engineering): software processes, software requirements and specifications, software design, testing and validation, tools and environments, and project management;
- *OPS* (Operating Systems): operating system function and design, concurrency, device management, security and protection, and file systems;
- *CSY* (Circuits & Systems): electrical quantities, resistive and reactive elements, frequency analysis, sinusoidal analysis, convolution, discrete time signals, filters, and Laplace and Z transforms;
- *NWK* (Networks): communication networks architecture and protocols, local and wide area networks, web as an example of client–server computing, data security, and wireless and mobile computing;
- *ELE* (Electronics): transistors, logic families, storage elements, interfaces and buses, op amps, amplifiers, filters, and integrated circuit (IC) building blocks;
- *DIG* (Digital Logic): switching theory, combinational logic circuits, memory elements, sequential circuit design, register transfer logic, and digital systems design;
- *PRF* (Programming Fundamentals): fundamental programming constructs, problem solving and data structures, programming paradigms, recursion, event-driven and concurrent programming, and using application program interfaces (APIs);
- *ALG* (Algorithms): basic algorithmic analysis, strategies, computability theory, complexity classes, and distributed algorithm;
- *ESY* (Embedded Systems): fundamentals, language issues, mapping between languages and hardware, real-time operating system (OS), tool support, and examples;
- *HCI* (Human Computer Interaction): devices and displays, static and exhibiting motion, and interaction with users—increasing sophistication;
- *INS* (Intelligent Systems): location awareness, determining awareness and utilizing it in devices, and aspects of intelligence, including learning;
- *INM* (Information Management): configuration management and version control and managing information in different contexts.

The following constitutes the knowledge areas that are currently under discussion and/or which appear to contribute optional content rather than core content:

- *DSP* (Digital Signal Processing);
- *VLS* (VLSI/ASIC Design);
- *DGA* (Design Automation);
- *ACP* (Alternative Computing Paradigms);
- *TFT* (Testing and Fault Tolerance);
- *LAC* (Language Considerations).

V. Technology Trends

It is relevant to look at current levels of performance in high-performance systems. As a reminder

$$\begin{aligned}\text{MB} &= 1\,000\,000 \text{ B.}\\ \text{GB} &= 1000 \ \text{ MB.}\\ \text{TB} &= 1000 \ \text{ GB.}\\ \text{Gigaflop} &= 10**9 \ \text{ instructions (floating point}\\ &\quad \text{operations) per second.}\\ \text{Teraflop} &= 1000 \ \text{ Gigaflops.}\\ \text{Petaflop} &= 10**15 \ \text{ instructions.}\end{aligned}$$

1) Communications: Networks that operate at the rate of 10 Gb/s currently exist. Gilder's law states that "*the total bandwidth of communication systems will triple every 12 months";* thus, by 2013, the capacity of communications systems will have moved to about 100 TB/s.

2) Computer Performance: There currently exist computers that carry out tens of teraflops per second. In August 2002, Fujitsu announced their HPC2500 high-performance computer, which contained sixteen 384 processors and achieved a performance of 85 teraflops/s. A commonly used predictor of developments is Moore's law, which implies that "*the processing power of a chip doubles every 18 months.*" It is widely believed that this law will remain valid until at least about 2020. When one looks at the implications of such performance, Moore's law suggests that performances are expected to rise to 1 petaflop/s by 2010 and 10**16 instructions/s by 2013. Computers currently exist that have a storage capacity of 10 TB of memory and 700 TB of disk space. Such capacity is again within the realms of Moore's law.

3) Value of a Network: Metcalfe's law is felt to govern the value of a network. It states that the value of a network is proportional to the square of the number of nodes in that network.

VI. Important Emerging Technical Areas

There are a number of technical areas that seem to be emerging and point to future developments in computing and computer engineering in particular.

A. Developments of the Internet

1) The Semantic Web: At the present time, material derived from the web is essentially in text format. The computer systems have no understanding of meaning. Consequently, they cannot seek to use the information to deduce information or to combine pieces of information from different sources to derive new information.

The concept of the semantic web in rough terms addresses this inability. Associated with information is its semantics or meaning. If computer systems are able to address questions of semantics, the route becomes open for systems to engage in interesting exchanges and to carry out deduction. Currently, these developments are some way off.

2) The Grid: The Internet can be viewed as a resource that makes readily available to everyone through online access enormous amounts of information of different kinds. The concept of the Grid is regarded as the next stage in this kind of development. Essentially, this infrastructure will provide various kinds of computing power as well as an information infrastructure and associated networking capability that will support many aspects of future activities of research, science, government, industry, etc.

This Grid is under development, currently and is being heralded as the successor to the Internet and, in many ways, a significant development beyond the Internet. Khosla described it basically as follows [20]

> "The Grid infrastructure will provide us with the ability to dynamically link together resources as an ensemble to support the execution of large-scale, resource-intensive, and distributed applications."

Many of the ideas contained in this document are taken from [19].

B. Pervasive Computing

The term "pervasive computing"—sometimes also called ubiquitous computing—has emerged as a result of the miniaturization whereby computers and computing devices are becoming extremely small. It was introduced in 1991 by Weiser (see [14]). Terms such as "the disappearing computer," "wearable computing," and "smart dust" have been coined to reflect similar phenomena. In its current incarnation, pervasive computing tends to have implications related to embedded systems (with all sorts of imaginative possibilities existing), smart badges as well as smart cars, smart buildings, and so on. More generally, however, the concept of pervasive computing gives rise to a number of concepts—thus, for example, the disappearing computer and wearable computing. Pervasive systems, to be effective and acceptable, need to be developed in such a way that they are minimally intrusive, i.e., they take account of aspects of the environment and use this information to ensure that they do not distract at awkward times. Hence, the concept of context aware systems becomes important. Of course, many issues of a professional and ethical nature emerge in the process.

C. Context-Aware Computing

1) Context Awareness: Deploying computer systems in the form of wearable computing presents many new challenges. These tend to stem from the observation that, unless designed carefully, a user can experience an avalanche of information that turns out to be a distraction at crucial times.

- There are new challenges for user interface design with context awareness being employed in many cases, especially in mobile environments.
- There should be a priority on not distracting and not infuriating the user; in this pursuit, some understanding of context is often again desirable; in general terms to become acceptable, pervasive systems need to be "minimally intrusive"; and to achieve this, context awareness is a priority.

The notion of "context" merits some attention [13], [16]. Typically in a computing context, this attention is to be interpreted to include location, personal history, and medical condition (including heart rate, body temperature, psychological state, daily behavioral patterns, and current situation (e.g., current task). There are implications in these observations for the methods, devices, etc., used to obtain this information in a manner that is unobtrusive. There are challenges relating to privacy and security, to where context information is stored, to how it can be arranged so that "information can be in the right place at the right time," to how this problem can be solved at minimum cost, to what are the fall-back positions in the event of the information not being available, to what are the appropriate technologies, and so on.

2) Content Adaptation: The concept of context awareness tends to be used in situations where humans are present. However, other possibilities exist. At a trivial level, of course, context awareness can be relevant not just to humans but, for instance, to robots, and medical devices of particular kinds. Consider a mobile situation in which a passenger in a car asks for the nearest garage. Depending on traffic density, it might be more sensible to go to a different garage since this action would be quicker and less expensive. Of course, such a situation suggests the presence of a decision engine that takes into account a whole spectrum of factors, many of which are related to context awareness. Such systems are referred to as *content adaptation systems.*

D. Adaptive Workplaces

Tied in with these earlier ideas are notions of adaptation [17], described as follows:

> "The word adaptive suggests that the workplace will change as external forces act on it. That is precisely the idea behind adaptive workplaces. As individuals are presented with tasks or transactions, generate their own ideas or perform creative work, the adaptive workspace will pick up on cues within the digital environment to automatically provide an end user with appropriate data and tools to perform their task."

One of the implications of this is that the system should not rely solely on predefined rules but should somehow leap beyond such restrictions to achieve the desired goals. The intention is not to be able to react to events that are implausible, but rather, to react to events that are plausible. There are two factors.

- On the one hand is the availability of distributed systems of increasing complexity and the advent of pervasive systems.
- On the other is the inability of transaction-oriented systems to respond properly to exceptions and the increased complexity of the work place.

The pressures for development toward adaptive work spaces become apparent. The technologies that would appear to support these developments [17] include: pattern recognition, automated classification methods, identity management, context awareness, collaborative methods, workflow and business rules, portals, and application integration. Many of these ideas are finding expression in the concept of *recommender systems* [16]. Such systems "*learn about user preferences over time, automatically finding things of similar interest*." An important effect of the existence of such systems is that the user is not frequently quizzed about issues. Rather, almost as a side effect of other activity, the system learns and continues to learn and adapt accordingly, making use of the ontology of information derived from other activity. This situation results in the giving of advice or recommendations.

Within [18], the comment is made that these systems will place new demands on operating systems that will increasingly be required to have the following characteristics:

- be self-organizing so that the needs of an individual in a particular situation can be addressed;
- be self-referential so that the system is aware of its environment and its own behavior and can react as appropriate;
- be adaptive so that it can change in the light of circumstances;
- be collaborative so that it can work with people or indeed with other systems;
- be anticipatory so that it seeks to look ahead and prepare for forthcoming eventualities.

The implication is for systems that are in some sense autonomous; they can plan to reorient themselves to meet perceived needs and perceived situations.

E. Autonomic Computing

One of the concerns about the possibility of pervasive computing is the set of issues that emerge from the occurrence of faults. To combat the possibility of mayhem, the idea of autonomic computing (a term coined by IBM) is to develop systems that are self healing, self-modifying, self-organizing, etc. In effect, they are self-managing and, thus, autonomous in some sense. It is natural to see autonomous computing as a development of the concept of fault tolerance. This concept applies to hardware systems, software systems, and information and information systems.

VII. CC2001—Computer Engineering

Within this volume, there is recognition that there are many possible interpretations of the term *computer engineering*. The implication is that there is scope for a range of different courses spanning these possibilities. A significant challenge is to ask about the underpinning ideas and whether these are likely to alter in the context of these developments. The fundamental concepts underpinning the computer engineering volume should remain unaltered.

Given the supposed developments outlined in Section VI, there is merit in asking whether the proposed curriculum could evolve to address the matters raised previously. In detail, the following lists the areas of development, and these are followed by the knowledge areas that should further evolve to encompass the proposed developments:

- *semantic web:* intelligent systems and networks;
- *grid development:* computer architecture and organization, high-performance computing, very large scale integration (VLSI) design, information management, and networks;
- *pervasive/ubiquitous computing:* computer systems engineering, computer architecture and organization,

operating systems, networks, and human–computer interaction;
- *context-aware computing:* intelligent systems, information management, human–computer interaction, networks, and computer architecture and organization;
- *adaptive systems:* intelligent systems, information management, networks, and human–computer interface;
- *autonomic computing:* fault tolerance, computer architecture and organization, and networks.

Within the description of the Computer Engineering volume, it would appear that all the elements necessary are in place. Of course, to remain current, in each case, the separate knowledge areas will need to evolve with technical advances over the years ahead. There is likely to be a shift in emphasis across the knowledge units. These same developments, of course, suggest a new focus and possible new titles for new degrees in the future. Of course, one important final observation is that the history of computing in general is littered with surprises.

VIII. Getting Involved

As the task force works toward completing the Computer Engineering Curriculum Report, it is extremely important for each person interested in computer engineering education in any way to get involved in the process. Needed are many public comments, criticisms, suggestions, and recommendations on the various drafts. The authors encourage the readers to contact the task force to get involved as a reviewer for the drafts (the website is at http://www.eng.auburn.edu/ece/CCCE). The curriculum report would be more productive and useful when it is backed up by the widest possible public consensus.

References

[1] "Computer Science in Electrical Engineering," COSINE Committee, Commission on Engineering Education, Washington, DC, Sept. 1967.

[2] L. A. Zadeh, "Computer science as a discipline," *J. Eng. Educ.*, vol. 58, no. 8, pp. 913–916, Apr. 1968.

[3] ACM Curriculum Committee on Computer Science, "Curriculum'68: Recommendations for academic programs in computer science," *Commun. ACM*, pp. 151–197, Mar. 1968.

[4] M. C. Mulder, "Model curricula for four-year computer science and engineering programs: Bridging the tar pit," *Computer*, vol. 8, no. 12, pp. 28–33, Dec. 1975.

[5] "A Curriculum in Computer Science and Engineering," Educ. Committee IEEE Comput. Soc., IEEE Publication EHO119-8, Jan. 1977.

[6] R. Austing, B. Barnes, D. Bonnette, G. Engel, and G. Stokes, "Curriculum'78: Recommendations for the undergraduate program in computer science," *Commun. ACM*, pp. 147–166, Mar. 1979.

[7] "The 1983 IEEE Computer Society Model Program in Computer Science and Engineering," IEEE Comput. Soc. Educ. Activities Board, Comput. Soc. Order No. 932, Dec. 1983.

[8] "Design Education in Computer Science and Engineering," IEEE Comput. Soc. Educ. Activities Board, Comput. So. Order No. 971, Oct. 1986.

[9] P. J. Denning, D. E. Comer, D. Gries, M. C. Mulder, A. B. Tucker, A. J. Turner, and P. R. Young, *Computing as a Discipline*. New York: ACM, 1988.

[10] Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force. [Online]. Available: http://computer.org/educate/cc1991/

[11] Computing Curricula 2001: Report of the ACM/IEEE-Computer Science Joint Curriculum Task Force. [Online]. Available: http://computer.org/educate/cc2001/

[12] *IEEE Pervasive Computing*, vol. 1, Oct.–Dec. 2002.

[13] M. Satyanarayanan, "Challenges in implementing a context-aware system," *IEEE Pervasive Computing*, vol. 1, no. 3, p. 2, July–Sept. 2002.

[14] M. Weiser, "The computer for the twenty-first century," *Sci. Amer.*, vol. 265, no. 3, pp. 66–75, Sept. 1991.

[15] S. E. Middleton, H. Alani, and D. C. De Roure, "Exploiting synergy between ontologies and recommender systems," presented at the Semantic Web Workshop, Hawaii, 2002.

[16] T. Selker and W. Burleson, "Context-aware design and interaction in computer systems," *IBM Syst. J.*, vol. 39, no. 3/4, 2000.

[17] D. W. Rasmus, *Adaptive Workplaces: Preparing for the Future of Work*: Giga Information Group, 2001.

[18] ——, "From Data Processing to Information Management: The Need for Intelligent Infrastructure," unpublished, 2002.

[19] F. Bermann, G. Fox, and T. Hey, Eds., *Grid Computing—Making the Global Infrastructure a Reality*. New York: Wiley , 2002.

[20] S. W. Khosla, P. K. Rohrer, and R. A. Rutenbar, "Reengineering and curriculum: Design and analysis of a new undergraduate electrical and computer engineering degree at Carnegie Mellon University," *Proc. IEEE*, vol. 83, pp. 1246–1269, Sept. 1995.

Andrew McGettrick was responsible for organizing and carrying out the assessment of the teaching quality of computing in all of the Scottish universities during the mid-1990s. In 2000, he chaired the group that created the U.K. benchmarking standards for computing and is now wrestling with benchmarking for the Master's provision. From 2002 to 2003, he has been providing advice to government agencies on the development of a software strategy for Scotland. He is currently Head of the recently formed Department of Computer and Information Sciences at the University of Strathclyde, Glasgow, U.K. Over the years he has had several research grants from research councils and the European Commission, and he is currently involved in a European Commission-funded project investigating the difficult issue of trust, with a particular focus on computing systems that involve mobile software. He has wide experience in editing (more than 120 books) and has published more than 130 papers and reports. His research interests are in the more formal aspects of software engineering

Dr. McGettrick is a Member of the Association for Computing Machinery (ACM) and the IEEE Computer Society and a Fellow of the Royal Society of Edinburgh (RSE), the British Computer Society (BCS), and the Institution of Electrical Engineers (IEE). Since 2001, he has been a Member of the Education Board of the ACM, and he is currently a Member of the joint ACM/IEEE-CS CC2001 Task Force and was involved in its Computer Science volume that is now published but has an ongoing role with the Computer Engineering, the Software Engineering, and the Overview volumes. In his activities with U.K. professional bodies, he has been Chairman of the Safety Critical Systems Committee (SCS) of the IEE (1996–present).

Mitchell D. Theys (S'90–M'99) received the B.S. degree in computer and electrical engineering, the M.S. degree in electrical engineering, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in 1993, 1996, and 1999, respectively.

He is currently an Assistant Professor in the Computer Science Department, University of Illinois at Chicago. His current research interests include distributed computing, heterogeneous computing, parallel processing, very large system integration (VLSI) design, and computer architecture. He has published several journal papers and numerous conference papers. He has received support from the Defense Advanced Research Projects Agency (DARPA), Intel, Microsoft, and the Armed Forces Communications and Electronics Association (AFCEA).

Dr. Theys is a Member of the IEEE Computer Society, Eta Kappa Nu, and Tau Beta Pi.

David L. Soldan (S'68–M'69–SM'84–F'01) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Kansas State University, Manhattan, in 1969, 1976, and 1980, respectively.

Dr. Soldan has worked in the areas of digital signal processing and adaptive filtering, computer networking, digital systems testing, computer systems reliability, manufacturing automation, and wireless communications. Over the years, he was formerly on the faculty at Oklahoma State University, Stillwater, and worked in industry for the NCR Corporation, Motorola, and Collins Radio Company. He served in the United States Air Force for four years and has served as a consultant to IBM, Frontier Engineering, ETO, and several universities. He is currently Professor and has been Head of Electrical and Computer Engineering at Kansas State University since 1989.

Dr. Soldan has served on the IEEE Education Society AdCom and has been the IEEE Computer Society Representative to the Frontiers in Education (FIE) Conference Steering Committee. He served as FIE Co-Program Chair in both 1995 and 1998 and served as President of the Electrical and Computer Engineering Department Heads Association from 2002 to 2003. He currently chairs the Computer Engineering Curriculum Committee of the IEEE Computer Society Computing Curriculum Taskforce. As a Member of the IEEE Committee on Engineering Accreditation Activities from 1999 to 2003, he was active in new program evaluator training and new evaluator mentoring.

Pradip K. Srimani (M'87–SM'90–F'99) received the Ph.D. degree in computer science from the University of Calcutta, Calcutta, India, in 1978.

He has served on the faculty of the Indian Statistical Institute, Calcutta, India; Gesselschaft fuer Mathematik und Datenverarbeitung, Bonn, Germany; the Indian Institute of Management, Calcutta, India; Southern Illinois University, Carbondale; and Colorado State University, Ft. Collins. Since 2000, he has been a Professor and Chair of Computer Science at Clemson University, Clemson, SC. He as been Guest Editor of special issues for many publications, including *Parallel Computing*; *Software*; the *Journal of Computer & Software Engineering*, the *Journal of Systems Software*; *VLSI Design*; and the *International Journal of Systems Science*. His research interests include reliable systems, parallel algorithms, fault-tolerant computing, networks, and graph theory applications.

He is a Member of the Association for Computing Machinery (ACM). He has served as past Editor-in-Chief of the IEEE Computer Society Press and is a Member of the Editorial Boards of the IEEE SOFTWARE MAGAZINE and the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. He has served as a Distinguished Visiting Speaker and Chapter Tutorial Speaker for the IEEE Computer Society for the past several years. He has been Guest Editor of special issues for IEEE publications, including the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, and the IEEE COMPUTER. He has also served many conferences in various capacities.