An integrated model program based on a 13-subject-area core curriculum provides academic planners a guideline adaptable to a range of options.

The IEEE Computer Society Model Program in Computer Science and Engineering

J. T. Cain, University of Pittsburgh
G. G. Langdon, Jr., IBM Research
M. R. Varanasi, University of South Florida

The computer field has undergone rapid technological changes over the last 30-35 years. During this time a well-established academic discipline has evolved. This discipline has various names, but will be referred to here as computer science and engineering. Model curricula recommendations for four-year undergraduate programs in the computer area have been reported in the past by the IEEE Computer Society,[1] the Association of Computing Machinery,[2] and the Cosine Committee.[3] These reports have been widely distributed and used by educators from many parts of the world.

The rapid pace of technological developments creates a need to update the CSE curricula in this field much more frequently than most other fields. In late 1981, for instance, the Educational Activities Board of the Computer Society undertook to update the 1977 model curricula. These rapid technological developments, however, have secondary effects on other aspects of a CSE program besides the curriculum. Namely student enrollments have increased, further straining program faculty and resource aspects of a program. In early 1982 the scope of the project was expanded to encompass all aspects of a program.

Overall goals

A number of institutions have complete CSE programs. In some cases, programs have been established in independent departments, while in other cases new programs have been established within existing departments and operate in parallel with the host program. In some institutions, only partial CSE programs exist as options of programs such as electrical engineering. In 1978, the Accreditation Board for Engineering and Technology established accreditation criteria for programs in the computer area. ABET criteria establish minimal, acceptable levels for the various aspects of an academic program. The ABET criteria are stated in a very general manner to allow for a wide variety of acceptable programs. However, the general nature of these stated criteria is also a cause of misinterpretation as to what the criteria actually require. Currently, there are over 25 accredited CSE programs, but many more universities are expected to request accreditation.

In the case of complete programs, the consensus opinion of a national group from the academic, industrial, and government sectors of the profession as to the breadth and depth of the computer area would be valuable viewpoints that the faculty could integrate with their own expertise and local constraints in revising and upgrading their curriculum. Established independent CSE programs normally have resources commensurate with other engineering and science programs at the institution. However, the rapid pace of developments in the computer field places severe strains on laboratory resources. Keeping equipment resources current is a problem that may be more severe in CSE programs than in other engineering and science programs.

While partial programs that are options within electrical engineering or other programs share many of the needs of complete programs, they usually have their own unique set of problems. In many cases, the faculty responsible for the option will have their degrees and most of their professional experience in the field of the main program (in most cases electrical engineering). Their expertise may not completely span the breadth of the CSE field, consequently the option curriculum will not be comprehensive. Naturally,

0018-9162/84/0400-0008$01.00

the option shares the resources of the host program. In many cases, this exacerbates the problem of expanding and keeping equipment and other resources current.

Very new or projected programs also have their own unique needs. In many cases, there are initially few or no faculty and/or administrators with expertise in the field. Hence, the expertise to develop a curriculum in the area may initially be missing, and the original faculty and administrators may lack knowledge of the faculty expertise and resources needed to establish a CSE program.

Recognizing these problems and needs, the committee decided to address all aspects of a CSE program—curriculum, faculty, and resources—and established the following overall goals for the model program in computer science and engineering project:

- Provide an overview of the desirable features of undergraduate academic programs in computer science and engineering;
- Provide a standard of comparison to guide the development of new programs or modify and upgrade established programs;
- Provide an interpretation of ABET criteria for minimum program standards;
- Establish a set of standards that can be used to define "Target of Excellence" programs;
- Define the CSE aspects of a curriculum in a manner that allows flexibility to meet the requirements of individual institutions;
- Provide guidance to academic administrators concerning the level of commitment needed to support a program.

A report of the results of the project is currently available from the Computer Society Press.* The following sections summarize the results presented in the report and the review process used to reach the consensus represented by the final report.

Curriculum

Model curricula reported in the past were developed for semester-based courses and treated only the CSE aspects of curricula.[1-3] Although this format has several advantages, it fails when attempts are made to integrate portions of the model into an existing curriculum at specific institutions. First, an institution may be on a quarter system, so at least a semester-to-quarter transformation of the model curricula is necessary. Then, under the quarter or semester system, local institution, college, school, or department requirements may make direct implementation of the model courses difficult. Even without these impairments, recommendations in the form of a fixed sequence of courses tend to reduce the creativity that the faculty should exercise in creating a curriculum customized to a particular institution.

In the early stages of the model program project, committee members eschewed identifying a single complete curriculum spanning four years. They preferred instead to identify a body of knowledge that should be a part of any curriculum. The criteria for selecting the fundamental concepts—the core of the curriculum—were to provide a student with broad background in engineering principles coupled with in-depth knowledge of hardware, software, application trade-offs and the basic modeling techniques used to represent the computing process. Another criterion for the core was that it stay within a 33-semester-credit-hour limit.

The Model Program Committee set the following goals for the curriculum aspect of the project.

- The curriculum would be specified in the form of subject areas rather than courses. A *subject area* is a collection of concepts arranged into modules that form a complete treatment of a specific technical area. Then implementations of subject areas can range from a portion of one or several courses to a sequence of complete courses. From the grouping of subject areas many different curricula can be generated.
- Subject areas would be developed to span the mainstream of computer science and engineering but would not be developed in interdisciplinary research areas.
- The committee would identify a set of core subject areas, which would constitute the fundamental concepts spanning the field and which should be the basis of any curriculum.
- In the non-CSE aspect of the curriculum (math, natural sciences, humanities, etc), the ABET criteria would be elucidated.
- Several sample implementations of complete four-year curricula would be included to illustrate the flexibility of arranging the subject into a complete curriculum.

The complete committee report details the core subject areas, the advanced subject areas (additional material that supplements the core for in-depth study of specific topics), and the non-CSE aspects of the curriculum. The following sections summarize important curriculum aspects of the report.

The core subject areas. The core of the model program consists of the following subject areas:

Core Lecture/Recitation Component

(1) Fundamentals of Computing
(2) Data Structures
(3) System Software and Software Engineering
(4) Computing Languages
(5) Operating Systems
(6) Logic Design
(7) Digital Systems Design
(8) Computer Architecture
(9) Interfacing and Communication

Laboratory Component

(10) Introduction to Computing Laboratory
(11) Software Engineering Laboratory
(12) Digital Systems Design Laboratory
(13) Project Laboratory

The subject areas are designed to provide a balance between hardware and software concepts through lecture/recitation classes reinforced by experimentation and

**The Model Program in Computer Science and Engineering* can be obtained through the CS west coast office, 10662 Los Vaqueros Cir., Los Alamitos, CA 90720; members, $10; nonmembers, $20; plus $2 handling.

Table 1. A sample core subject area—logic design (SA6). This subject area covers the digital building blocks, tools, and techniques in digital design. A building-block approach to logic design is emphasized in this segment of instruction. This segment requires approximately three semester hours.

Module	Topic	Purpose	Prerequisites	Corequisites	Concepts	References*
1	Introduction to Logic Circuits	To introduce the student to gate-level logic circuit analysis and design. Fundamental characteristics of relevant electronic technologies should be introduced along with the concepts of propagation delay, fan-in, fan-out, and power dissipation and noise margin.	—	Basic Electronics	Basic logic elements Technologies and levels of integration (TTL, MOS, CMOS, ECL) Boolean algebra and switching theory Representation and manipulation of switching functions Realization of switching functions Minimization of switching functions	4-12
2	Combinational Logic Circuits	To introduce the student to small-and medium-scale integrated circuits as building blocks. Particular emphasis should be given to design of combinational logic networks using the most commonly employed SSI and MSI circuits.	Logic Design, Module 1	—	Multiplexers and demultiplexers Decoders and encoders Adders and subtracters Carry look-ahead and carry completion techniques Selective (controlled) inverters Comparators Programmable logic arrays (PLAs) and read-only memories (ROMs) Hierarchical nature of functional units Designing with MSI Combinational circuit propagation delays, combinational hazard	4-7, 9, 11, 13, 14
3	Memory Elements	To introduce the student to basic memory elements and to provide an insight into timing considerations. Clocking, control and timing considerations should be thoroughly discussed.	Logic Design, Module 2	—	Fundamental (unclocked) mode versus clocked mode circuits Basic flip-flops SR, JK, D-latch, D-triggered Flip-flop clocking, control and timing Level-sensitive devices, edge-triggered devices, master-slave devices Setup, propagation, and hold times Data registers Selection and clocking Timing characteristics	4-7, 9, 11, 13, 15, 16
4	Sequential Logic Circuits	To introduce analysis and synthesis of sequential circuits.	Logic Design, Module 3	—	Finite state machines Clocked machines and unclocked machines Descriptive techniques State diagrams and state tables Algorithmic state machine charts Analysis of synchronous and asynchronous circuits Design of synchronous sequential circuits State minimization State assignment Next state equation realization Sequential functional units Shift registers, counters, sequence detectors, synchronizers, debouncers, controllers Designing with MSI	4-7, 9-11, 14, 16
5	Register Transfer Logic	To introduce the notion of register transfer as a higher-level way to describe system behavior.	Logic Design, Module 3	—	Register transfer notation Data flow components Control flow components Conditional and unconditional transfers ALU control Bus structures Logic sequencers Programmable PLAs	4, 5, 9, 17-19

*Please see the reference list at the end of this article.

projects in the laboratory component. (See sample core subject area in Table 1.) A substantial laboratory component is included to refine problem-solving techniques through experimentation. The laboratory material is structured to make the laboratory an environment where students "learn by doing." The intent of the project laboratory is to integrate the concepts in hardware and software through well-chosen applications.

Within the subject areas, the prerequisite structure is defined at a module level to facilitate many different implementations; see Figure 1, which illustrates the structure of a subject area.

Advanced subject areas. Advanced subject areas supplement the core for in-depth study in specific topics. Implementation of some of these subject areas may range from a portion of one or several courses to a sequence of complete courses, so core subject areas are repeated as advanced subject areas.

No institution is expected to be able to include all of these areas in a curriculum. However, a program must include a sufficient number of courses from related subject areas to cover two or more areas of in depth study. The list of subject areas is not exhaustive: Specific programs may include other subject areas.

The subject areas proposed in the 1983 model program go considerably beyond those of the 1977 report.[1] One area, however, microprogramming, no longer appears as a distinct subject area: It is a design alternative for the control unit of a CPU in subject area 7. New areas include software engineering, fault-tolerant computing, performance, graphics, VLSI design, and artificial intelligence.

The advanced subject areas listed in the report are

(14) Software Engineering
(15) Digital Design Automation
(16) Therory of Computing
(17) Database Systems
(18) Advanced Computer Architecture
(19) Design and Analysis of Algorithms
(20) Fault-Tolerant Computing
(21) Performance Prediction and Analysis
(22) Computer Graphics
(23) VLSI System Design
(24) Translator Writing Systems
(25) Computer Communications Networks
(26) Systems Laboratory
(27) Artificial Intelligence
(28) Advanced Operating Systems

A sample advanced subject area is given in Table 2.

Sample implementation. This is a semester-based program consisting of 134 semester credit hours (see Table 3). The implementation is an ideal without the institutional constraints that would prevent early coverage of key core material or sufficient flexibility for a variety of areas of specialization.

The professional electives of Table 4 are selected to develop an area of concentration in one of the CSE subfields. The following electives are typical, but the actual selection will be a function of the current interests of the faculty associated with the program. Note in Table 5 that compilers and translators, operating systems, and architecture appear in each subfield area and are actually required professional electives; only the time when they are

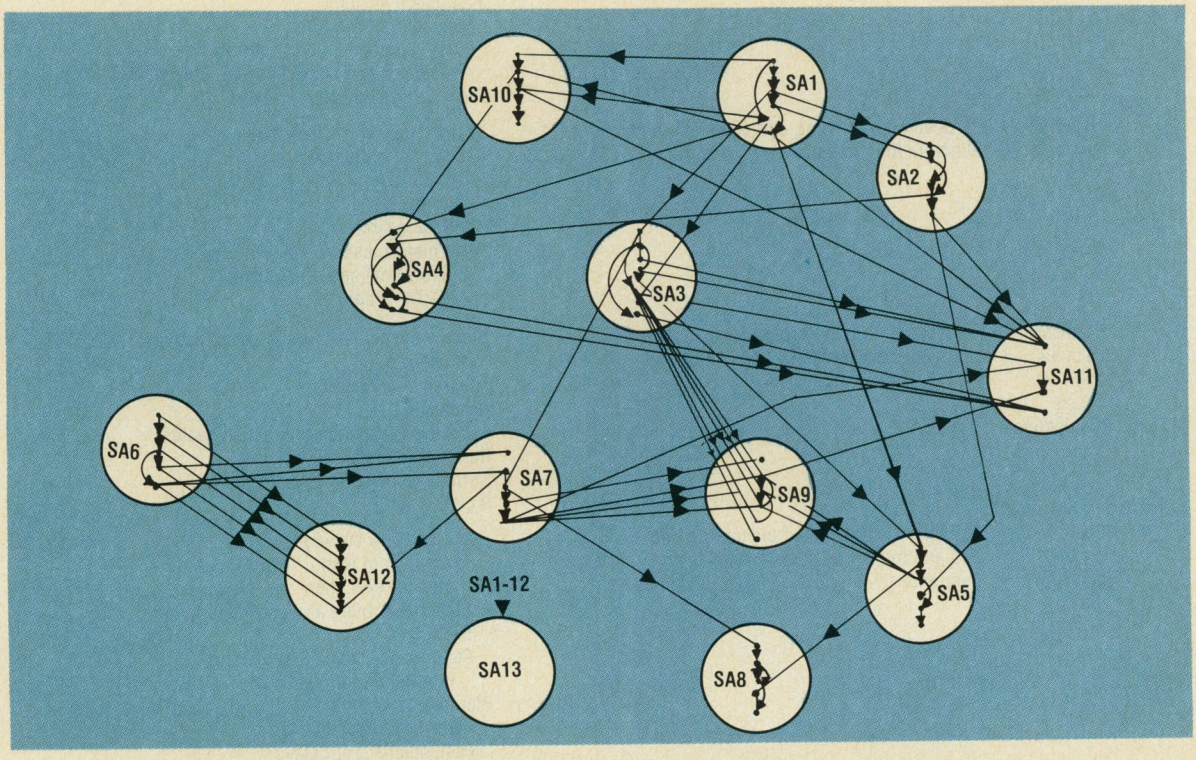


Figure 1. Prerequisite relations among subject areas in the core. Each dot represents a module within a subject area. Directed segments represent the prerequisite structure.

taken is elective. The other electives are considered free electives and the student should be able to select courses from a wide range of areas to complete a solid program of study (Table 6).

Faculty

Existing, fully developed programs have established faculties whose expertise spans, as a minimum, the fundamental core of the CSE area. In addition, the research interests of individual faculty members provide several curriculum areas in which a student can pursue in-depth study. New or projected programs and few partial-option programs have a core faculty with expertise spanning the CSE area. When the three classes of programs are faced with recruiting a new faculty member, there is sometimes a tendency by the administration to "convert" faculty from other areas into CSE faculty. Generally, conversion re-

Table 2. A sample advanced subject area—SA27, artificial intelligence. The objective of these instruction segments is to provide an overview of (1) how computers can be applied to solve problems and (2) the principles of human intelligence.

Module	Topic	Purpose	Prerequisite	Concepts	References
1	Intro. to problem solving with computers	To introduce the student to the general subject of artificial intelligence (AI).	SAs 1-5 and 8	Examples of: learning, geometric analogies, simple understanding, problem solving, integral calculus programs, robotics, expert consulting	20, 21
2	Intro. to representations	To introduce the elementary notion of representation.	SA27, Module 1	Procedural—computer programs Production systems Nonprocedural First-order predicate calculus Assertion collections Semantic networks Descriptive tools Differences Constraints Description-based example Constraint-based sentence analysis Prototypes, scripts, schemata	20-26
3	Control strategies	To introduce the elementary notion of control.	SA27, Module 2	Means-ends analysis or general problem solver Problem decomposition Propagation of constraints: Huffman-Clowes labeling Production systems Situation-action rules Data, operations, and control Decomposition or partitioned subsystems Rule-based deduction systems	20, 22, 24, 25, 27, 28
4	Searching strategies	To describe searching strategies and to illustrate their importance.	SA27, Module 3	Search trees and exhaustive search Backtracking methods Hill-climbing Graph-search Heuristic search Performance issues Bidirectional and unidirectional AND/OR graphs	20, 22, 24, 28, 29
5	Predicate calculus and rule-based deductions	To describe the first-order predicate calculus and its associated uses within the field of Artificial Intelligence.	SA27, Module 3	Syntax, concepts, formulas, connectives, quantifiers Theorems and proofs Resolution Resolution refutation production system Rule-based deduction systems Structured object representations Prolog	20, 22, 24, 27
6	Goal-directed planning	To explain goal-directed planning strategies and to provide some examples and illustrations.	SA27, Module 3	Robot problem solving Perception of world, to form plan, to monitor execution State description and goal description STRIPS Deduction system, Green's and Kowalski's formulations Hierarchical planning Hierarchy of conditions Postponement of details Patch higher-level plans from lower level details	20, 22, 24

quires an extensive period of formal retraining. Unfortunately, in many cases, conversions are imposed without formal retraining.

The Model Program Committee set the following goals for the faculty aspect of the project.

- Identify the minimal administrative structure necessary for a viable program.
- Identify the educational and/or professional background that potential faculty members should possess.
- Specify the minimum full-time equivalent faculty necessary for a viable program irrespective of student body size or nature of the institution.
- Address the issue of part-time faculty and set guidelines for maximum percentages, proper recruitment, and proper recognition of part-time faculty.

Table 2. (continued)

Module	Topic	Purpose	Prerequisite	Concepts	References
7	Applications of understanding	To illustrate the notion of understanding in the contexts of image analysis and language.	SA27, Module 3	Image analysis Pattern recognition Scene description, primal sketches Computer vision, constraints Language understanding Limited domains of discourse Conceptual dependency Noun groups Transition network Handling questions and commands	20, 21, 26, 30
8	Representation of knowledge: frames and scripts	To provide a detailed examination of frames, and to illustrate the notion with an augmented transition network and the analysis of grammars	SA27, Modules 3 and 7	Networks and frames Nodes, relations, terminals, markers Prerequisite conditions, viewpoint transformations Defaults, expectations, information retrieval Augmented transition network Case grammars Scripts	20, 22, 23, 25, 26, 28, 31
9	Programming languages and databases for artificial intelligence	To present the importance of the proper programming language as a way of viewing problems, to introduce a language (LISP, Prolog, or others), and to introduce the notions of pattern-oriented databases and demons.	SA27, Modules 3 and 7	Relationship of programming to AI Symbol and manipulation problems Programming languages for AI Need for special languages Lisp and Prolog Example, Lisp program for Augmented Transition Network Pattern-oriented databases Associative databases Application of demons or monitors	20, 22, 27, 28
10	Knowledge-based systems	To unify concepts of earlier modules in the context of applications that require a large body of knowledge, i.e., knowledge-based (expert) systems, as opposed to clever computational procedures. Case studies of knowledge-based systems are presented.	SA27, Modules 3, 7, and 9	Review relevant knowledge representation concepts Knowledge involved, how much, what is it, domain-specific FRL, KRL, KL-ONE Review simple turtle programs Data, knowledge base, control State, input, subgoal, feedback, iteration, recursion, debugging Knowledge acquisition Knowledge-based systems: case studies Dendral and meta-dendral Hearsay II Consultation system MYCIN, Prospector Knowledge transfer system: TEIRESIAS Automatic programming system: PECOS	20-23, 25-28, 30, 32-34
11	Robotic systems	To study the application of CSE and AI for development of robotic systems.	SA27, Module 7	Robotic arms Robotic vision Automated manufacturing Languages for robotic systems Tele-operator systems	35-37

Table 3. Model curricula recommendations.

Student's Year	First Semester Course	Units	Second Semester Course	Units
Freshman	Chem. 101 (Chemistry 1)	4	Physics 101 (Physics 1)	4
	Math 101 (Calculus 1)	4	Math 102 (Calculus 2)	4
	CSE 101 (Intro. to Computing 1)	3	CSE 102 (Intro to Computing 2)	3
	English 101 (English Comp.)	3	Humanities/Social Sciences (Electives)	6
	Humanities/Social Sciences (Electives)	3		
		Total = 17		Total = 17
Sophomore	Physics 201 (Physics 2)	4	Science Elective	4
	Math 201 (Calculus 3)	3	Math 202 (Differential Equations)	4
	CSE 201 (Intro. Comp. Eng. 1 Hardware & Software Design)	2	CSE 203 (Intro. Hardware Design Lab)	3
			EE 201 (Basic Circuits & Electronics)	3
	CSE 202 (Discrete Structures)	3	Humanities/Social Sciences (Electives)	3
	Humanities/Social Sciences (Electives)	3		
		Total = 16		Total = 17
Junior	CSE 301 (Hardware System Design Lab)	3	CSE 303 (Software Eng.)	3
	CSE 302 (Software Eng. & Assembler Programming)	4	Engineering Science (Mechanics)	3
		3	Prof. Elective (see Table 4.)	6
	Engineering Science (Linear systems Analysis)	3	Elective	3
		3		Total = 15
		Total = 16		
Senior	CSE 401 (Design Lab)	3	CSE 402 (Design Lab)	3
	Prof. Elective (see table 4.)	12 or 9	Prof. Elective (see Table 4.)	9 or 12
	Elective	3 or 6	Elective	6 or 3
		Total = 18		Total = 18

The committee agreed that a strong CSE program requires a faculty committed to the program and leader by a program chairman with the authority to make the major decisions that influence the operation and growth of the program. At least five permanent full-time faculty members are needed for a viable program regardless of student enrollment; as the number of students increases, so must the faculty. Precise student faculty ratios are a function of whether there is a graduate/research program in addition to the undergraduate program.

Table 4. Professional electives by subfield area.

Student's Year	Professional Elective	Software Engineering	Computer System Design	Knowledge Based Systems
Junior	1	Compilers and Translators	Architecture	Database Systems
	2	Intro. to Performance Analysis	Advanced Electronics	Theory of Computing
Senior				
	3	Operating Systems	Computer Communication Networks	Operating Systems
	4	Database Systems	Operating Systems	Artificial Intelligence
	5	Theory of Computing	Compilers & Translators	Architecture
	6	Translator Writing Systems	Design Automation	Compilers and Translators
	7	Architecture	Intro. to Performance Analysis	Computer Communication Networks
	8	Computer Communication Networks	VLSI System Design	Distributed Databases
	9	Design and Analysis of Algorithms	Fault-Tolerant Computing	Graphics

Entry-level faculty members should normally have a PhD in CSE. For senior positions, the record of achievement in the computer area is more important than the field of doctoral study, since many senior members entered the field before CSE curricula existed. In addition, the program must offer faculty members an opportunity to engage in research, development, or other professional opportunities.

Part-time faculty can be a valuable resource to a program by contributing valuable insight to the students as well as by covering a portion of the teaching load. However, no more than 20 percent of the teaching load should be covered by part-time faculty. Special attention should be paid to recruitment, coordination, and proper recognition for the contributions of part-time faculty.

Resources

The newness of the academic field and the rapid pace of technological development in the area create a unique set of CSE resource needs. Many institutions planning to implement a CSE program have little appreciation of the resources necessary for a viable program. The continual

advances in both hardware and software make the job of keeping laboratory and computing resources current even more difficult than for the other fields of engineering and science.

The rapid growth in the number of textbooks, journals, conference proceedings, etc., creates further problems—keeping library aspects of the resources up to date. The committee set the following resource goals for the project.

- Identify the computing resource requirements for a viable program in CSE.
- Identify the laboratory resources necessary for a program.
- Determine the library resources necessary to support a program.
- Determine the general support requirements that are necessary to have viable program operation.

The committee reviewed the central computing resource requirements from the viewpoints of changes in technology, the type of software support required by CSE programs, and the number, type, and performance characteristics of the I/O devices. Members underscored the importance of high-speed, interactive facilities. They identified laboratory resource requirements on the basis of personnel, hardware, software, and support staff. They insisted that the laboratory incorporate a substantial design component and emphasized the vital role that faculty should play in the laboratories and how their efforts should be recognized.

Committee members also stipulated that programs include professional librarians in order to have a viable library and ensure the type of services that should be provided. They also identified secretary-staff-office and support equipment needs.

Review procedure

Many CSE specialists have contributed to this model program. The report took its final form through the generous volunteer effort of both the contributors and the reviewers. Reviews accompanied all phases of the preparation. The Model Program Committee held open meetings at Compcons, Compsacs, and National Computer Conferences throughout 1982 and 1983. There were also several presentations on various aspects of the recommended program at educational conferences. As a result, the specification of the subject areas and the noncurricular recommendations benefitted from a broad input.

The initial set of subject areas for the core was identified, developed, and written by summer 1982. Then, core subject areas were distributed for review to approximately 50 experts from academia, industry, and government. A revised core was prepared on the basis of the reviews received. After this initial review, the advanced subject areas and noncurricular aspects of the program were developed. As each section was drafted, it was individually reviewed by one or more knowledgeable professionals, then revised.

By Compsac 1982, a complete draft report was available. It was circulated to the Computer Society Educational Activities Board, Executive Committee, and Governing Board for review and comment, as was a revised draft at Compcon Spring 1983.

In June 1983, the committee distributed a complete draft together with some basic review guidelines to approximately 300 reviewers. They were selected for their particular expertise. A random sample of deans of colleges of engineering and deans of colleges of arts and sciences provided guidance regarding the general usefulness of the report from an administrative perspective. The chairpersons of all ABET-accredited computer science and engineering programs were asked to provide reactions to the overall program requirements. The chairpersons of all Computer Society Technical Committees, a subset of the Educational Activities Board mailing list, and additional individuals received copies primarily to give their reactions to specific areas within the report. Approximately 50 responses were received to these mailings.

This set of reviews had an overall positive tone and supported completion and release of the report in substantially the form of the June 1983 draft. Nonetheless, comments and suggestions received were cross-referenced to the report and presented to the Model Program Committee. Although not all suggested changes caused a revision in the final document, they were all carefully considered. The analysis and review were completed in early fall 1983, with appropriate revisions included in the final report.

Table 5. Mapping to model program core.

SA	Course Title	Course Numbers
1	Fundamentals of Computing	CSE 101, 102
2	Data Structures	CSE 101, 102, 201
3	System Software & Software Engineering	CSE 302
4	Computing Languages	'Required' Prof. Elec.
5	Operating Systems	'Required' Prof. Elec.
6	Logic Design	CSE 102, 201, 203
7	Digital Systems Design	CSE 201, 203, 301
8	Computer Architecture	'Required' Prof. Elec.
9	Interfacing and Communications	CSE 203, 301, 302
10	Introduction to Computing Lab	CSE 101, 102
11	Software Laboratory	CSE 303
12	Digital Systems Design Lab	CSE 203
13	Project Lab	CSE 401, 402

Table 6. The ABET requirements are exceeded in each category. This is typical of most actual implementations.

Requirement	Subject	Course Numbers
1/2 year	Mathematics	Math 101, 102, 201, 202; Stat 301; CSE 202
1/2 year	Basic Sciences	Chem 101; Phys 101, 102; Basic Science Electives
1/2 year	Humanities/Social Sciences	Humanities/Social Sciences Electives
1 Year	Engineering Science	EE 201, 301; ES 301; CSE 101 102, 301; Professional Electives, 27 credits
1/2 year	Engineering Design	CSE 101, 102, 201, 202, 301, 303, 401, 402

The final revisions were not major, perhaps because of the extensive earlier reviews. Only subject areas 3, 5, and 11 of the core material required major revision, although numerous additional materials and suggestions were incorporated in other areas. The sample implementation, Tables 3, 4, 5 and 6, was rewritten to be more representative of a CSE curriculum with strong CSE content. Among other changes resulting from this review were the inclusion of a graphical representation of the prerequisite structure of the program, Figure 1. Many reviewers commented that the modules did not represent uniform amounts of teaching time, and requested that weights be assigned. The committee felt that the weight attached to the concepts, be it an in-depth treatment or only the definition of the concept, was a choice each instructor should make depending on local conditions. However, the report now contains estimates of the time required to teach the subject areas of the core. Responding to reasonable requests for certain new core material, the committee increased the semester-credit-hour limit for the core curriculum to 33 semester hours, although 30 credits was an early goal. Reviewer comments also caused a number of modifications in the faculty and resource aspects of the program.

The model program presented here was developed by a committee of the Educational Activities Board of the IEEE Computer Society. The curriculum component of the program is intended to provide potential graduates with a well-balanced education in fundamental principles of hardware and software design, reinforced with experiential skills. It is organized into subject areas for ease of implementation into courses either on a quarter or semester basis. Further, the curriculum is divided into the core material and the advanced material. The core material consists of concepts that provide the students with a thorough understanding of software and hardware design as well as laboratory experiments which give them experience in integrated system design. The advanced material spans the breadth of the computer area in theory, software engineering, VLSI design and computer applications in communications, and areas such as robotics. The sample implementation illustrates a method of not only meeting, but going well beyond the ABET accreditation requirements for our field.

The document also addresses resources, such as faculty, laboratories, and other support, which are needed for programs in computer science and engineering. With a commitment to resources, any institution that so desires can use this report to custom design a program that suits its particular needs and the expertise of its faculty. *

Acknowledgments

This article summarizes the results of the model program project. It could not have been accomplished without the hard work and dedication of many people. We thank all those who originated, stimulated, and continued the project: the coordinators who oversaw the various aspects of the project; the contributors who wrote modules or subject areas or who made other organizational contributions; and the reviewers of the various drafts of this document.

The Model Program task was begun in 1981 under C. V. Ramamoorthy, then vice president of the Computer Society for Educational Activities, and continued and expanded by his successor, Taylor Booth. The Model Program Committee consisted of Bill Carroll, Ron Danielson, Jerry Engel, Lee Hollaar, Pei Hsia, Rao Vemuri, Ben Wah, and the authors. Additional contributors and reviewers are mentioned by name in the foreword of the document itself. We also thank the east coast office of the IEEE Computer Society for its assistance.

References

1. *Model Curricula in Computer Science and Engineering*, Education Committee of the IEEE Computer Society, report no. EH0119-8, IEEE-CS Press, Los Alamitos, CA, Jan., 1977.
2. *ACM Recommended Curricula for Computer Science and Information Processing Programs in Colleges and Universities, 1968-1981*, Committee on Computer Curricula of the ACM Education Board, ACM Press, New York, 1981.

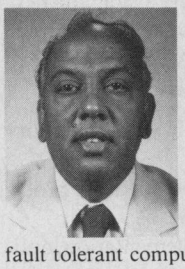
3. *Computer Sciences in Electrical Engineering,* Cosine Committee, Commission on Engineering Education, National Academy of Engineering, Washington, DC, 1967.
4. T. L. Booth, *Introduction to Computer Engineering: Hardware and Software Design,* John Wiley & Sons, New York, 1984.
5. D. L. Dietmeyer, *Logic Design of Digital Systems,* Allyn and Bacon, Rockleigh, NJ, 1978.
6. W. I. Fletcher, *An Engineering Approach to Digital Design,* Prentice-Hall, Englewood Cliffs, NJ, 1980.
7. F. J. Hill and G. R. Peterson, *Introduction to Switching Theory and Logical Design,* John Wiley & Sons, New York, 1981.
8. G. G. Langdon, Jr., "A Building Block Approach to Digital Design," *IEEE Potentials,* Vol. 2, No. 2, 1983, pp. 7-10.
9. M. Mano, *Digital Logic and Computer Design,* Prentice-Hall, Englewood Cliffs, NJ, 1979.
10. H. T. Nagel, B. Carroll, and J. Irwin, *An Introduction to Computer Logic,* Prentice-Hall, Englewood Cliffs, NJ, 1975.
11. John Peatman, *Digital Hardware Design,* McGraw-Hill, Hightstown, NJ, 1980.
12. G. Williams, *Digital Technology,* SRA, Palo Alto, CA, 1981.
13. T. R. Blakeslee, *Digital Design with Standard MSI and LSI,* Wiley-Interscience, New York, 1979.
14. D. Winkel and F. Prosser, *The Art of Digital Design,* Prentice-Hall, Englewood Cliffs, NJ, 1980.
15. T. Bartee, *Digital Computer Fundamentals,* McGraw-Hill, Hightstown, NJ, 1981.
16. C. Wiatrowski and C. House, *Logic Circuits and Microcomputer Systems,* McGraw-Hill, Hightstown, NJ, 1980.
17. J. Hayes, *Computer Architecture and Organization,* McGraw-Hill, Hightstown, NJ, 1978.
18. G. G. Langdon, Jr., *Computer Design,* Computeach Press, San Jose, CA, 1982.
19. M. Sloan, *Computer Hardware and Organization,* SRA, Palo Alto, CA, 1983.
20. *Handbook of Artificial Intelligence,* A. Barr, P. R. Cohen, and E. A. Feigenbaum, eds., Kaufmann, Los Altos, CA, Vols. 1-3, 1981, 1982.
21. P. H. Winston, *Artificial Intelligence,* Addison-Wesley, Reading, MA, 1977.
22. R. Kowalski, *Logic for Problem Solving,* Elsevier-North Holland, New York, 1979.
23. G. McCalla and N. Cercone, eds., "Approaches to Knowledge Representation," *Computer,* Vol. 16, No. 10, Oct. 1983.
24. N. Nilsson, *Principles of Artificial Intelligence,* Tioga Press, Palo Alto, CA, 1980.
25. R. C. Schank and C. K. Riesbeck, *Inside Computer Understanding: Five Programs Plus Miniatures,* L. Erlbaum Assoc., Hillsdale, NJ, 1981.
26. J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine,* Addison-Wesley, Reading, MA, 1984.
27. W. F. Clocksin, and C. S. Mellion, *Programming in Prolog,* Springer-Verlag, New York, 1982.
28. P. H. Winston, "Learning Structural Descriptions from Examples," in *The Psychology of Computer Vision,* P. Winston, ed., McGraw-Hill, Hightstown, NJ, 1975.
29. E. Horowitz and Sartaj Sahni, *Fundamentals of Computer Algorithms,* IEEE-CS Press, Los Alamitos, CA, 1978.
30. *Reading in Artificial Intelligence,* B. L. Webber and N. J. Nilsson, eds., Tioga Publishing, Palo Alto, CA, 1981.
31. *Representation and Understanding,* D. G. Bobrow and A. Collins, eds., Academic Press, New York, 1975.
32. *Building Expert Systems,* F. Hayes-Roth, D. Waterman, and D. Lenat, eds., Addison-Wesley, Reading, MA, 1983.
33. *Expert Systems in the Microelectronics Age,* D. Michie, ed., Edinburgh Univ. Press, Edinburgh, 1979.
34. D. S. Nau, "Expert Computer Systems," *Computer,* Vol. 16, No. 2, Feb. 1983, pp. 63-85.
35. K. S. Fu, "Robotics and Automation," *Computer,* Vol. 15, No. 12, Dec. 1982, pp. 34-40.
36. R. A. Jarvis, "A Computer Vision and Robotics Laboratory," *Computer,* Vol. 15, No. 6, June 1982, pp. 8-24.
37. S. S. Reddi and E. A. Feustel, "A Restructurable Computer System," *IEEE Trans. Computers,* Vol. C-27, No. 1, Jan. 1978, pp. 1-20.

J. T. Cain is a guest editor of this issue. His photo and biography appear on p. 6.

G. G. Langdon, Jr., is a senior member of the IEEE Computer Society and is cochairman of the Model Program Committee of the Educational Activities Board.

With IBM since 1963, Langdon is interested in computer design and architecture, mapping algorithms to VLSI, error detection, and data compression systems. At the University of Sao Paulo, 1971-1972, Langdon's students designed and built Brazil's first digital computer.

Langdon received his BSEE from Washington State University, his MSEE from the University of Pittsburgh, and his PhD in EE from Syracuse University.



M. R. Varanasi is an active member of the IEEE Computer Society and is currently involved in developing a model program in Computer Science/Engineering for the Educational Activities Board of the IEEE Computer Society.

Currently an associate professor in the department of Computer Science and Engineering at the University of South Florida, his research interests include coding theory, fault tolerant computing, computer architecture and image processing. From 1973 to 1980 he was with the department of electrical engineering, Old Dominion University, Norfolk, VA.

Varanasi received his BSC and DMIT degree from Andhra University and Madras Institute of Technology, India, and his MS and PhD degrees in electrical engineering from the University of Maryland, College Park, in 1972 and 1973 respectively.

Questions about this article can be directed to J. T. Cain, Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA 15261.