The ACM and IEEE-CS Guidelines for Undergraduate CS Education

David Hemmendinger
Department of Computer Science
Union College
Schenectady, NY 12308
+1 518 388-6319
hemmendd@union.edu

INTRODUCTION

The ACM has provided curricular recommendations for computer science (CS) since its 1965 preliminary report [A1]. There have been four iterations of its guidelines, 1968 [A2], 1978 [A3], 1991 [A5], and 2001 [A6]. The field is still rapidly changing, but there is also some consensus about its core. Nonetheless, the curricular guidelines display significant variation in focus and in emphasis. Two persistent issues have been the role of mathematics and the importance of programming in the introductory courses. Another has been the importance of new application areas. This paper reviews the history of these curricular guidelines.

BACKGROUND

There were several studies of computing and curricula in the early 1960s, including ones by the National Academy of Sciences [A14], the Mathematical Association of America [A9], and Tompkins [A12]. These addressed the definition of computer science, academic computing needs, computing in the mathematics curriculum, and the name of computing programs. The 1963 ACM National Conference had a panel discussion on CS curricula, leading to papers in a curriculum issue of CACM, sponsored by the ACM Education Committee and edited by Bernard Galler. In its lead article, Keenan [8] characterized the central subject matter of computer science as

> ... the study of the organizational and structural properties of systems, arrays of symbols and mechanical languages which find their application in the processing and communication of information .... (p. 206)

He listed four areas with which computer science is concerned:

1. Organization of equipment constituting an information processing system (including both people and machines)
2. Software systems that control and communicate with equipment
3. Procedures and theories for specifying computational processes
4. Applications of all of these to other disciplines

Keenan describes five areas of CS education: general education, programmer training, computing for scientists, systems programming, and computer science for researchers and teachers. The paper does not describe curricula, but is followed by descriptions of introductory CS courses by Bruce Arden (Michigan) [2] and by Alan Perlis (Carnegie Institute of Technology) [9]. Each is ambitious, addressing algorithms, algorithmic languages, machines — both abstract and concrete, and the problems of implementing algorithms in programs that run on machines with finite representations of numbers.

The first ACM curricular recommendations came from the Curriculum Committee on Computer Science (C^3S) in 1965: "An undergraduate program in computer science: preliminary recommendations" [A1]. The committee began as a subcommittee of the Education Committee in 1962 and

I thank Tony Ralston, David Wise, and the anonymous referees for their helpful comments.

became independent two years later. The report's definition of computer science abstracts away some of the detail in Keenan's definition:

> Computer science is concerned with *information* in the same sense that physics is concerned with energy; it is devoted to the *representation, storage, manipulation* and *presentation* of information in an environment permitting automatic information systems.

Its proposed curriculum was intended to prepare students for graduate work either in CS or in other fields, for systems programming, and for applications programming, and to this end it had a small core and a set of electives (Table 1). This core was supplemented by a five-course math requirement (calculus, analysis, linear algebra), with more math highly recommended, and with electives in related fields such as electronics.

	basic	theory	numerical algorithms	computer models and applications
required	intro to algorithmic processes comp. organization info. structures	algorithmic languages and compilers	numerical calculus	
highly recommended electives	logic design and switching theory computer and programming systems		numerical analysis I numerical analysis II	
advanced electives	combinatorics and graph theory	constructive logic automata theory formal languages		systems simulation optimization techniques heuristic programming

Table 1: Curriculum of the "Preliminary Report"

Numerical computation is a major part of this curriculum, although it is the topic of only one of the five basic courses, which generally make algorithmic thinking central, as do the Perlis and Arden introductions. The elective "constructive logic" course includes part of what we now call "discrete mathematics" — sets, propositional and predicate logic, algorithms.

CURRICULA 68 AND 78

Curriculum 68

The Curriculum 68 (C68) report noted that the "Preliminary recommendations" addressed the definition and justification of CS as a discipline, and that there was continuing debate about the name of the field [A8]. The 1968 report accepted the name "computer science" and organized the subject into three areas, reflecting the Keenan definition: "Information structures and processes" included much of the core of CS now: data structures, programming languages, computational models. "Information processing systems" covered the gamut from low-level hardware to major system components and operating systems. "Methodologies" included major application areas such as AI, numerical methods, graphics, and simulation.

Like the preliminary report, C68 emphasized algorithmic thinking in its first course, saying that the notion of an algorithm should be clearly distinguished from that of a program. The report called attention to two courses that were not in the earlier report. One was computer organization, now separated from assembly-language programming. Although the report does not describe it in this way, the effect of the separation is to permit the treatment of both software and

hardware as the means of implementing algorithms.

The other "particularly notable" basic course was on discrete structures. It included some of what was in the earlier constructive logic course together with graph theory, but most important, moved down to the elementary level as a required course. Table 2 contains the full list of courses in the curriculum.

C68 had a strong mathematical emphasis through its requirements of traditional courses in mathematical analysis, its several courses on numerical computation, and its electives in formal languages and computation theory. Its orientation appeared to reflect the origin of many CS programs in departments of mathematics or engineering rather than in business programs. A *Datamation* report comments, "The committee's viewpoint may be entirely correct.... Freezing out the business administration student seems somewhat arbitrary, however" [1]. Pollack's survey of the state of computer science in 1982 [10] describes it as having "a dichotomizing aspect: Its basically mathematical orientation sharpened its contrast with more pragmatic alternatives" (p. 41). He added that not only was there continuing debate about the appropriate introductory course; the report was also criticized for downplaying application areas and professional preparation in fields like commercial data processing.

In 1971 Gerald Engel surveyed 26 institutions offering a Ph.D in computer or information science to learn how much of C68 they had implemented [6]. Twenty-two to 25 departments offered each of the eight required courses or a similar one, or had such a course in another department (e.g. discrete structures in a math department). Advanced courses offered by the fewest programs were information retrieval, information processing systems, graphics, simulation, sequential machines, and hybrid computing.

Only 14 of the institutions had undergraduate programs; all offered the first basic course, twelve offered the second, and ten offered programming languages. Seven to nine programs offered each of the remaining core courses. Of the other courses in Table 2, I5, I6, I8 were offered by four programs, and the rest, by one or two each. To judge from this one small sample, overall programs matched the curriculum surprisingly well, but less well at the undergraduate level, perhaps because much of the material was not yet considered suited to that level.

Curriculum 78 (C78)

The C^3S committee met in 1974 to consider revisions of its curriculum, and preliminary reports appeared in the SIGCSE Bulletin later that year. In 1977, it published "A survey of the literature in computer science education since Curriculum '68" [A4]. Engel's report on the meeting [7] lists some areas of discussion, and comments, "Of note was agreement that basic computer science is fundamentally the study of programming languages to implement data structures in the environment of hardware" (p. 80). Although one should not read too much into a single sentence, it is noteworthy that programming languages are at the center here and that algorithms are not mentioned. He goes on to say that the rest of the undergraduate program becomes "tracks off of this 'heart'", and that meeting participants agreed that "no mathematics beyond high school was required for this heart."

Curriculum 78 [A3] had an eight-course core: two on programming; three on computer systems, organization, and architecture; data structures and algorithms, file processing, and programming languages. The complete list of required and elective courses is in Table 2. The report lists six objectives of a CS major. Three include being able to write programs, assess them, and knowing how to use appropriate tools; the others are to understand basic computer architecture, work alone or in a team, and be able to pursue further education in the field.

Where C68 emphasized algorithms, information structures and models of computation in its core, C78 emphasized programming skills and its core included an application area, file processing.

Many of the upper-level courses remained similar, and the new curriculum was notable in recommending a course on computers and society.

	Curriculum 68		Curriculum 78	
	computer science	math	computer science	math
Basic core	B1 intro to computing	calculus	CS1 programming I	*required, non-core:*
	B2 computers and programming	analysis I	CS2 programming II	calculus
			CS3 comp. systems	analysis I
	B3 discrete structures	analysis II	CS4 comp. organization	lin. algebra
	B4 numerical calculus	linear algebra	CS5 file processing	probability
				discrete structures
Intermediate core	I1 data structures	probability	CS6 OS & comp. arch I	
	I2 prog. languages		CS7 data structures and algorithm analysis	*elective:*
	I3 comp. organization			analysis II
	I4 systems programming		CS8 programming language organization	prob. & stat.
	I5 switching theory*	adv. calculus*		
	I6 sequential mach.*	alg. structures*		
	I7 num. analysis I*	probability &		
	I8 num. analysis II*	statistics*	*four required, no more than two in any subfield:*	
Advanced	A1 formal languages		CS9 computers and society	
	A2 advanced computer organization		CS10 OS and computer architecture II	
	A3 analog and hybrid computing		CS11 database management systems	
	A4 systems simulation		CS 12 AI	
	A5 information retrieval		CS13 algorithms	
	A6 computer graphics		CS14 software design	
	A7 theory of computability		CS15 programming language theory	
	A8 large-scale information systems		CS16 automata, languages, computability	
	A9 AI, heuristic programming		CS17 numerical math: analysis	
			CS18 numerical math: linear algebra	
	* two each of CS and math required		plus advanced topics courses	

Table 2: Curriculum 68 and Curriculum 78

Another prominent feature was that although the mathematics courses they required were similar, the discrete-structures course moved from the CS core to a more advanced mathematics course in C78, one that was a prerequisite for only a single advanced course on automata and computability. In fact, not only was no math course required for the core, as Engel had said, the only other CS courses that depended on any of them were the two on numerical methods. This lack of connection between mathematics and the computing curriculum was criticized by Ralston and Shaw [11], who argued for a two-year discrete-math sequence to accompany the core.

COMPUTING CURRICULA 1991 and 2001

1980s

Curricular discussions during the 1980s took two directions. In 1981, the IEEE Computer Society and the ACM started what became the Computer Science Accreditation Board (CSAB), charged with establishing requirements for professional certification in computer science that were similar to those of the Accreditation Board for Engineering and Technology (ABET) for computer engineering. Its requirements included about a year and a half of computer science, a year of science and mathematics, and addressed matters such as laboratories and number of faculty. The first CSAB accreditations were made in 1986.

During the same time there were also criticisms of C78, most notably by a group of faculty from liberal-arts colleges, funded by the Alfred P. Sloan Foundation. The group, now known as the Liberal Arts Computer Science (LACS) consortium, met in 1984 and 1985 and in 1986 published its model curriculum [A10]. It argued that C78 was obsolete, having failed "to explicate adequately the principles that underlie the discipline" (p. 203) — and indeed, to have failed to incorporate these principles except in advanced electives. It also sought to define a relatively small curriculum that could be offered by a small department.

The theme of this model curriculum was that computer science *is* science. It proposed a six-course core: CS1-2 (algorithms, programming, data structures) computer organization, algorithms, theory of computation, and programming languages, along with three electives, an early course on discrete math, and two more math courses. Its first two courses closely resembled the 1984-85 revised recommendations for those courses from the ACM Curriculum Task Forces for CS1 and for CS211, [A12]. with one exception: it called for CS2 to provide an overview of computer science to orient students toward further study.

Computing Curricula 1991

In 1985 the ACM, with the cooperation of the IEEE Computer Society, formed a Task Force on the Core of Computer Science, with the goal of "describ[ing] its intellectual substance in a new and compelling way" [4, p. 9], one that would present it as more than programming. Its 1988 report, "Computing as a discipline," was intended to facilitate curricular discussion; a condensed version appeared in 1989 [4].

The report, addressing both computer science and engineering, sought a definition of the field that would be concrete, generally comprehensible, and a "rallying point"; would reflect its historical origins and present its fundamental questions and accomplishments:

> The discipline of computing is the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application. The fundamental question underlying all of computing is, "What can be (efficiently) automated?"

The report elaborated three "paradigms," theory, abstraction (modeling), and design, that play a role in all areas of the discipline, and enumerated nine subareas:

> algorithms and data structures
> programming languages
> architecture
> numerical and symbolic computation
> operating systems
> software methodology and engineering
> databases and information retrieval
> artificial intelligence and robotics
> human-computer interaction

The Curricula 91 (CC91) report adopted this two-dimensional organization, which has the virtue of permitting a relatively small set of courses to address core topics and to treat them both theoretically and practically. Its opening section listed the influences of previous reports and its responses. Among others, these included:

"Computing as a discipline": a breadth-first approach in introductory courses.

CC91 promote breadth in several alternative ways, and has broader goals than that report.

1983 IEEE-CS model curriculum for computer science and engineering: organization into small units; sample implementations meeting accreditation guidelines.

CC91 addresses a broader range of institutions and programs.

Curriculum 78: prominent role of programming, detailed course descriptions; course on computers in society.

CC91 has more current subject matter, treats programming differently and better integrates theory, abstraction, design, and the social context of computing

CC91 had eleven subject areas (adding a programming-language introduction and social, ethical, and professional issues to the 1988 report's nine). It had three "processes" (the CC91 term for the 1988 report's paradigms) and it listed twelve recurring concepts that would help to unify a curriculum; some examples are binding, large-problem complexity, efficiency, abstraction levels, security. It broke each of the subject areas into core knowledge units (KUs), labeling each with some of the recurring concepts. The core was composed of 56 KUs, including an optional one to introduce a programming language, requiring 283 lecture hours. The distribution of KUs among the areas varied from two to twelve. CC91 also called for a half-year of mathematics, including discrete math and calculus. There was also a list of advanced elective topics, but the analysis into KUs didn't extend to the electives even when a title matched one of the nine areas.

The report presented twelve sample sets of courses to illustrate ways of composing the KUs — in computer science, computer engineering, each with a breadth-first variant, liberal-arts programs, and some others. Although these were quite thoroughly worked out, they proved to be a weak point. In particular, despite the call for breadth in the 1988 report, the LACS report and CC91, few breadth-first programs emerged.

The CC91 program was relatively heavy for a liberal-arts curriculum, and in 1996 the LACS group revised its model [A15] to adapt CC91. It had a 240-hour core, replaced its earlier course on computational theory by a plan to treat theory as a recurring theme, and added more formal laboratories to its CS1-2, as the IEEE-CS and ACM proposals had recommended. It reduced the CC91 core to 240 hours largely by giving less attention to architecture, operating systems, and software methodology, though it devoted more time to algorithms, including parallel computation.

Computing Curricula 2001: Computer Science (CC2001)

In view of the broadening scope of computing, the ACM and IEEE-CS Computing Curricula project initiated in 1998 planned a multi-volume set of curricula, five of which are now drafted or complete: Computer Engineering, Computer Science, Information Systems, Information Technology, and Software Engineering, and an overview report.

The CC2001 task force surveyed about 150 CS departments and found that CC91 appeared less influential than its predecessors for several reasons: many people found KUs less useful than actual course designs — the survey found that many schools still used the C78 courses; they wanted a smaller set of core topics; they wanted help with accreditation.

Among its design principles, CC2001 retained the theory-abstraction-design triad, embraced lifelong learning, sought to identify fundamental skills and knowledge while keeping the core as small as possible, and was intended to offer course-design guidance.

CC2001 divided CS into fourteen areas, similar to those of CC91, though rearranged and extended. It moved discrete structures from mathematics into the CS core (thereby returning to the C68 model). It added areas to the core that had become central in the previous decade, such as networks, graphics, and human-computer interaction, though some of these contributed only a few core hours. Its areas were broken into topics (like KUs), not all of which were in the required core of 280 course hours. Its 43 hours in discrete mathematics, a separate course in CC91, gave it the same number of core units as CC91 but an eighth fewer core hours. Since CC2001 analyzed its areas into topics that went beyond the core and did not propose that basic courses comprise solely core topics, it could help with course design more than CC91 did.

CC2001 has been praised for moving discrete mathematics into its core, in one or perhaps two courses. It has also been criticized, however, for its omission of topics that would draw on or reinforce the mathematics material; e.g. in programming languages [3]. A valuable part of the CC2001 report is its outline of multiple introductory sequences: imperative-first, objects-first, functional-first, breadth-first, algorithms-first, and hardware-first. Each shows how to partition important elementary topics into two- and three-course introductions. To date, some variation of one of the first two, closest to the traditional CS1-2, has been most common, along with a minority committed to functional-first. The difficulty in implementing a breadth-first introduction, in addition to a scarcity of textbooks, is that it requires some change to most of the curriculum and is thus difficult to introduce.

CONCLUSION

Curriculum 68 and 78 represent an initiative and a reaction, and Curricula 91 and 2001 are similarly related. In each case the first of the pair defined an approach to the CS curriculum based on reflection on the field — C68 had to establish the first comprehensive curriculum, and CC91 followed an attempt to characterize the discipline. Curriculum 78 was shaped by reactions against the mathematical emphasis of C68 and gave more weight to professional education and to applications. CC2001 remedied what educators saw as insufficient attention to course outlines in CC91, and modified its program to include the growth of areas like networking that had become fundamental to the field.

C78 helped to establish the "computing as programming" view that its successors have attempted to replace, with only modest success. CC2001 has a core program that is arguably more fragmented than CC91. Although both have some core areas with only two or three topics and a correspondingly small time alloted to them, CC2001 has its topics spread over more areas than the previous curriculum by virtue of its taking a broader view of what is central to computer science.

C68 had a significant influence on early programs. The influence of later guidelines is not so clear. C78 was arguably as much a reflection of how curricula had developed as it was an influence on them. CC91 proposed to found curricula on basic principles, and it led to some attempts to offer broad introductory courses, but what survived in CC2001 was the notion of constructing a curriculum from a set of small units that could be composed into courses in several ways. The latter emphasized concrete course models more than CC91 did, and proposed ways to integrate traditional and new areas in some of them.

From the time of the first ACM curriculum, there have been multiple curricular threads, not all addressed by the early curricula, as commentators on C68 noted. The recent CC2005 Overview Report [A7] addresses this diversity, maps the terrain and places computer science, computer engineering, software engineering, information systems, and information technology within it. It briefly characterizes the shared identity of these programs (pp. 35-36). As it notes, many of the shared elements vary with the discipline, such as "essential and foundational underpinings," whether they lie in formal theory or in practice. Among the more concrete are a) an understanding of programming: algorithms and data structures, programming skills, basic understand of hardware, and software design principles; and b) an understanding of what computing systems can and cannot do, both in principle and in current technology.

CC2005 does not propose a core curriculum to be shared by all of its programs, and that may not be possible in view of their diversity, though worth considering. As computing continues to broaden in scope, however, there will be a continuing need to identify the common elements. A recent article by Wing [13] proposes "computational thinking" as an attitude and set of skills that are universally applicable, and this notion might help to shape a curricular core of the computing disciplines.

The task of defining a core of a broad computer science curriculum is similar: to identify what every student of computer science needs to know — the most important concepts and skills, the ones essential to enabling a student to understand the discipline and continue to become educated after graduation. CC91 attempted to address this task with its three processes and twelve recurring concepts, but these are too many and the latter represent a mixture of many different levels.

The liberal-arts model CS curricula, including the 2004 one [A13], have been designed to be consistent with current ACM curricular guidelines. They have a small core with considerable depth in discrete math, algorithms, architecture, and programming languages, and with application areas left for elective choice. There have also been more radical proposals for a very "small footprint" for computer science [5]. It would be interesting to see if the next ACM curriculum could identify a small model for its core, adding advanced courses and courses on applications as needed. A curriculum focused sharply on a small number of fundamental concepts could also be an attractive basis for the variety of interdisciplinary programs that are now emerging.

References

1. (anon.), Conference report: Curriculum '68, *Datamation 14,* 5 (1968), 114-116.
2. Arden, B. W., On introducing digital computing, *Communications of the ACM 7,* 4 (Apr. 1964), 212-214.
3. Bruce, K. A., Curriculum 2001 draft found lacking in programming languages, Programming Languages Knowledge Area Focus Group for Curricula 2001, `http://www.cs.williams.edu/~kim/Curric2001/PLKFG.html`, 2001.
4. Denning, P. J., Comer, D. E., Gries, D., Mulder, M. C., Tucker, A. B., Turner, J. A., and Young, P. R., Computing as a discipline, *Communications of the ACM 32,* 1 (Jan. 1989), 9-23.
5. Downey, A. B. and Stein, L. A., Designing a small-footprint curriculum in computer science, *Proceedings of the 36th ASEE/IEEE Frontiers in Education Conference* (Oct. 2006), `http://fie.engrng.pitt.edu/fie2006/papers/1689.pdf`.
6. Engel, G. L., Input from ACM Curriculum Committee on Computer Science, *ACM SIGCSE Bulletin 3,* 4 (Dec. 1971), 30-39.
7. Engel, G. L., Initial report: the revision of "Curriculum 68", *ACM SIGCSE Bulletin 6,* 3 (Dec. 1971), 79-80.
8. Keenan, T. A., Computers and education, *Communications of the ACM 7,* 4 (Apr. 1964), 205-209.
9. Perlis, A. J., Programming of digital computers, *Communications of the ACM 7,* 4 (Apr. 1964), 210-211.
10. Pollack, S. V., The development of computer science, in *Studies in Computer Science,* ed. S. V. Pollack, MAA Studies in Mathematics *22,* 1-51, Mathematical Association of America, Washington, D.C., 1982.
11. Ralston, A. and Shaw, M., Curriculum '78 — Is computer science really that unmathematical?, *Communications of the ACM 23,* 2 (Feb. 1980), 67-70.
12. Tompkins, H. E., Computer education, in *Advances in Computers,* ed. F. L. Alt and M. Rubinoff *4,* 135-168, Academic Press, New York, 1963.
13. Wing, J. M., Computational thinking, *Communications of the ACM 49,* 3 (2006), 33-35.

APPENDICES FOR THE DIGITAL LIBRARY VERSION

Appendix I: Authors of the curriculum reports

Curriculum '68 (C^3S committee)

William F. Atchison	Univ. of Maryland, College Park
Samuel D. Conte	Purdue Univ., W. Lafayette, IN
John W. Hamblen	SREB and Georgia Institute of Technology, Atlanta
Thomas E. Hull	Univ. of Toronto, Toronto, Ont., Canada
Thomas A. Keenan	EDUCOM, and Univ. of Rochester, Rochester, NY
William B. Kehl	Univ. of California at Los Angeles, Los Angeles
Edward J. McCluskey	Stanford Univ., Stanford, CA
Silvio O. Navarro	Univ. of Kentucky
Werner C. Rheinboldt	Univ. of Maryland, College Park
Earl J. Schweppe	Univ. of Maryland, College Park
William Viavant	Univ. of Utah
David M. Young, Jr.	Univ. of Texas

Curriculum '78 (contributors to the report)

Richard Austing	University of Maryland (editor)
Bruce Barnes	National Science Foundation (editor)
Della T. Bonnette	University of Southwestern Louisiana (editor)
Gerald Engel	Old Dominion University (editor)
Gordon Stokes	Brigham Young University (editor)
Robert M. Aiken	University of Tennessee
Michael A. Arbib	University of Massachusetts
Julius A. Archibald	SUNY at Plattsburgh
William Atchison	University of Maryland
Victor R. Basili	University of Maryland
Barry Bateman	Southern Illinois University
W.P. Buckley	Aluminum Company of America
Frank Cable	Pennsylvania State University
Gary Carlson	Brigham Young University
B.F. Caviness	Rensselaer Polytechnic Institute
Donald Chand	Georgia State University
Sam Conte	Purdue University
William Cotterman	Georgia State University
Daniel Couger	University of Colorado
John F. Dalphin	Indiana University-Purdue University at Fort Wayne
Gene Davenport	John Wiley and Sons
Charles Davidson	University of Wisconsin
Peter Denning	Purdue University
Ed Desautels	University of Wisconsin
Benjamin Diamant	IBM
Karen A. Duncan	MITRE Corporation
Michael Faiman	University of Illinois
Patrick Fischer	Pennsylvania State University

Arthur Fleck	University of Iowa
John Gannon	University of Maryland
Norman Gibbs	College of William and Mary
Malcolm Gotterer	Florida International University
David Gries	Cornell University
H.C. Gyllstrom	Univac
Douglas H. Haden	New Mexico State University
John W. Hamblen	University of Missouri-Rolla
Preston Hammer	Grand Valley State Colleges
Richard Hamming	Naval Postgraduate School
Thomas R. Harbron	Anderson College
Stephen Hedetniemi	University of Oregon
Alex Hoffman	Texas Christian University
Charles Hughes	University of Tennessee
Lawrence Jehn	University of Dayton
Karl Karlstrom	Prentice-Hall
Thomas Keenan	National Science Foundation
Sister M.K. Keller	Clarke College
Douglas S. Kerr	The Ohio State University
Rob Kling	University of California, Irvine
Joyce C. Little	Community College of Baltimore
Donald Loveland	Duke University
Robert Mathis	Old Dominion University
Daniel McCracken	President, ACM
Robert McNaughton	Rensselaer Polytechnic Institute
M.A. Melkanoff	University of California, Los Angeles
John Metzner	University of Missouri-Rolla
Jack Minker	University of Maryland
Howard Morgan	University of Pennsylvania
Abbe Mowshowitz	University of British Columbia
Michael Mulder	Bonneville Power Administration
Anne E. Nieberding	Michigan State University
James Ortega	North Carolina State University
F.G. Pagan	Memorial University of Newfoundland
John L. Pfaltz	University of Virginia
James Powell	North Carolina State University
Vaughn Pratt	Massachusetts Institute of Technology
Anthony Ralston	SUNY at Buffalo
Jon Rickman	Northwest Missouri State College
David Rine	Western Illinois University
Jean Sammet	IBM
John F. Schrage	Indiana University-Purdue University at Fort Wayne
Earl Schweppe	University of Kansas
Sally Y. Sedelow	University of Kansas
Gary B. Shelly	Anaheim Publishing
James Snyder	University of Illinois
Theodor Sterling	Simon Fraser University
Alan Tucker	SUNY at Stony Brook
Ronald C. Turner	American Sign and Indicator Corporation

Brian W. Unger	The University of Calgary
James Vandergraft	University of Maryland
Peter Wegner	Brown University
Patrick Winston	Massachusetts Institute of Technology
Peter Worland	Gustavus Adolphus College
Marshall Yovits	The Ohio State University
Marvin Zelkowitz	University of Maryland

Computing Curricula 1991

Allen B. Tucker (Co-chair)
Bruce H. Barnes (Co-chair)
Robert M. Aiken
Keith Barker
Kim B. Bruce
J. Thomas Cain
Susan E. Conry
Gerald L. Engel
Richard G. Epstein
Doris K. Lidtke
Michael C. Mulder
Jean B. Rogers
Eugene H. Spafford
A. Joe Turner

Computing Curricula 2001

Vice-President, IEEE-CS Education Activities Board
Carl Chang, Vice-President, IEEE-CS Education Activities Board
Peter J. Denning Chair, ACM Education Board

IEEE-CS delegation
James H. Cross II (co-chair)
Gerald Engel (co-chair and editor)
Robert Sloan (secretary)
Doris Carver
Richard Eckhouse
Willis King
Francis Lau
Susan Mengel
Pradip Srimani

ACM delegation
Eric Roberts (co-chair and editor)
Russell Shackelford (co-chair)
Richard Austing
C. Fay Cover
Gordon Davies
Andrew McGettrick
G. Michael Schneider
Ursula Wolz

Appendix II: Additional References

A1. ACM Curriculum Committee on Computer Science, An undergraduate program in computer science — preliminary recommendations, *Communications of the ACM 8,* 9 (Sep. 1965), 543-552.

A2. ACM Curriculum Committee on Computer Science, Curriculum '68, *Communications of the ACM 11,* 3 (Mar. 1968), 151-197.

A3. ACM Curriculum Committee on Computer Science, Curriculum '78, *Communications of the ACM 22,* 3 (Mar. 1979), 147-166.

A4. ACM Curriculum Committee on Computer Science, A survey of the literature in computer science education since Curriculum '68, *Communications of the ACM 20,* 1 (Jan. 1977), 13-21.

A5. ACM/IEEE-CS Joint Curriculum Task Force, *Computing Curricula 1991* (1991), ACM Press and IEEE Computer Society Press.

A6. ACM/IEEE-CS Joint Task Force on Computing Curricula, *Computing Curricula 2001: Computer Science,* `http://www.acm.org/education/curricula.html`, Dec. 2001.

A7. ACM/IEEE-CS Joint Task Force on Computing Curricula, *Computing Curricula 2005: Overview Report,* `http://www.acm.org/education/curricula.html`, Sep. 2005.

A8. Atchison, W. F. and Hamblen, J. W., Status of computer sciences curricula in colleges and universities, *Communications of the ACM 7,* 4 (Apr. 1964), 225-227.

A9. Committee on the Undergraduate Program in Mathematics (CUPM), *Recommendations on the undergraduate mathematics program for work in computing.,* Mathematical Association of America (May 1964).

A10. Gibbs, N. E. and Tucker, A. B., A model curriculum for a liberal arts degree in computer science, *Communications of the ACM 29,* 3 (Mar. 1986), 202-210.

A11. Koffman, E. B., Miller, P. L., and Wardle, C. E., Recommended curriculum for CS1, 1984, *Communications of the ACM 27,* 10 (Oct. 1984), 998-1001.

A12. Koffman, E. B., Stemple, D., and Wardle, C. E., Recommended curriculum for CS2, 1984, *Communications of the ACM 28,* 8 (Aug. 1985), 815-818.

A13. Liberal Arts Computer Science Consortium, A 2004 model curriculum for a liberal arts degree in computer science, `http://www.lacs.edu/model-curriculum.pdf`, *Feb. 2004.*

A14. National Academy of Sciences, Committee on Computer Uses, *Computer needs in American colleges and universities,* Publ. Dept., NAS, Washington, D. C., 1964.

A15. Walker, H. M. and Schneider, G. M., A revised model curriculum for a liberal arts degree in computer science, *Communications of the ACM 39,* 12 (Dec. 1986), 85-95.