COMPUTING CURRICULUM - COMPUTER ENGINEERING (CCCE) A MODEL FOR COMPUTER ENGINEERING CURRICULA IN THE NEXT DECADE

Victor P. Nelson[1], David L. Soldan[2], Andrew McGettrick[3], John Impagliazzo[4], Pradip Srimani[5], Mitchell D. Theys[6] and Joseph L. A. Hughes[7]
[1]Auburn Univ./ [2]Kansas State Univ./ [3]Univ. of Strathclyde/ [4]Hofstra Univ./ [5]Clemson Univ./ [6]Univ. of Illinois at Chicago/ [7]Georgia Inst. of Technology

Abstract

In the fall of 1998, the Computer Society of the Institute for Electrical and Electronic Engineers (IEEE-CS) and the Association for Computing Machinery (ACM) established the Joint Task Force on Computing Curricula 2001 (CC2001) to undertake a major review of curriculum guidelines for undergraduate programs in computing. The effort was to match the latest developments of computing technologies in the past decade and last through the next decade [1]. The "Computing Curriculum 1991" [2] and other previous efforts of the IEEE-CS and ACM did not distinguish computer science from computer engineering programs. The IEEE-CS and ACM established the Computing Curriculum - Computer Engineering (CCCE) Task Force in 2001 to develop a separate volume on computer engineering curricula to complement the CC2001 report. Other task forces also emerged to prepare separate volumes for computer science, information systems, information technology, and software engineering.

The work of the CCCE Task Force appears as a report available for review on the web [3]. This report has undergone extensive review, including an NSF-sponsored workshop. By the time of this conference, the final report will have been presented to the IEEE-CS and ACM, and made available for distribution. This paper presents an overview of that report.

The Computing Curriculum - Computer Engineering (CCCE) Report

One must understand the nature of a discipline and its needs before designing a curriculum to produce graduates who can work effectively in that discipline. The CCCE report begins by discussing computer engineering as a discipline, including an overview of how the field of computer engineering has evolved, characteristics of computer engineering graduates, and the corresponding curricular preparation required to practice computer engineering.

Computer engineering embodies the science and technology of design, construction, implementation, and maintenance of software and hardware components of modern computing systems and computer-controlled equipment. Computer engineering has traditionally occupied the territory that lies at the interface between computer science and electrical engineering. It evolved over the past three decades as a separate, although intimately related, discipline. Computer engineering is solidly grounded in the theories and principles of computing, mathematics, science, and engineering and it applies these theories and principles to solve

Proceedings of the 2004 American Society for Engineering Education Annual Conference and Exposition Copyright © 2004, Amerian Society for Engineering Education

technical problems through the design of hardware, software, networks, and processes.

Increasingly, computer engineers are involved in the design of computer-based systems to address highly specialized and specific application needs. Computer engineers work in most industries, including the computer systems affecting aerospace, telecommunications, power production, manufacturing, defense, and electronics industries. They design high-tech devices ranging from tiny microelectronic integrated-circuit chips, to powerful systems that utilize those chips and efficient telecommunication systems that interconnect those systems. Technological advances and innovation continue to drive computer engineering. There is now a convergence of several established technologies, which has created many opportunities and challenges for computer engineers. This convergence of technologies and the associated innovation lie at the heart of economic development and the future of many organizations. The situation bodes well for a successful career in computer engineering.

The Computer Engineering body of knowledge

The most significant effort of the CCCE Task Force has been the identification and organization of the body of knowledge (BOK) pertinent to undergraduate study of computer engineering. Following the pattern of the Computer Science Volume [4], the CPE BOK is organized hierarchically into three levels. The highest level of the hierarchy is the knowledge *area* that represents a particular disciplinary subfield. Each area is identified by a three-letter abbreviation, such CAO for *Computer Architecture and Organization*. This identifier is prefixed with "CE-" to distinguish areas appearing in the Computer Engineering Report from similar areas in the Computer Science Report. As listed in Table 1, the BOK comprises eighteen knowledge areas; sixteen relate directly to computer engineering and two relate to mathematics (discrete structures, probability and statistics).

Table 1. Computer Engineering Body of Knowledge Areas

CE-ALG	Algorithms and Complexity
CE-CAO	Computer Architecture and Organization
CE-CSE	Computer Systems Engineering
CE-CSG	Circuits and Signals
CE-DBS	Database Systems
CE-DIG	Digital Logic
CE-DSP	Digital Signal Processing
CE-ELE	Electronics
CE-ESY	Embedded Systems
CE-HCI	Human-Computer Interaction
CE-NWK	Computer Networks
CE-OPS	Operating Systems
CE-PRF	Programming Fundamentals
CE-SPR	Social and Professional Issues
CE-SWE	Software Engineering
CE-VLS	VLSI Design and Fabrication
CE-DSC	Discrete Structures
CE-PRS	Probability and Statistics

The knowledge areas are broken down into smaller divisions called knowledge *units* that represent individual thematic modules within an area. A numeric suffix to the area name identifies each unit; as an example, CE-CAO3 is a (knowledge) unit on *memory system organization and architecture*. An appendix to this paper presents a complete list of BOK units. A set of learning objectives accompany each unit. Finally, each unit is further subdivided into a set of *topics* that are the lowest level of the hierarchy. These are presented in detail in Appendix A of the CCCE Report.

Core and elective units

One of the Task Force goals in proposing curricular recommendations was to keep the required component of the body of knowledge as small as possible to allow for program flexibility. To this end, the Task Force has defined a minimal *core*, comprising those units for which there is broad consensus that the corresponding material is essential to anyone obtaining an undergraduate degree in computer engineering. The BOK areas listed in Table 1 are those for which one or more units have been designated as core. Units taught as part of an undergraduate program, but which fall outside the core, are considered to be *elective*. The BOK areas listed in Table 1 contain a number of elective units. Some additional computer engineering areas from which one can select elective topics to supplement the core material appear in Chapter 7 of the CCCE Report. These additional areas are not exhaustive; other elective areas are certainly valid topics for computer engineering as new areas evolve continuously.

Several points should be noted. First, the organization of the BOK is not intended to suggest an organization of a curriculum or individual courses, although the report does include several sample implementations of these. Second, one would expect that an undergraduate program will cover the minimum core topics plus additional elective units from the body of knowledge, as illustrated in Figure 1. The core units nominally account for about one-fourth of a typical undergraduate computer engineering program. The CCCE report does not specify which elective units should be included. Computer engineering programs can have different program objectives and, as a result, will have different emphases. The breadth of a program at a particular institution will stipulate which elective knowledge units will complement the core areas. Thus, it is incumbent upon each local program to encompass as much of the body of knowledge as possible. Third, core units are not necessarily limited to introductory courses. Although many of the units defined as core are indeed introductory, some core units can appear only after students have developed significant background in the field. The designation *core* simply means *required* and says nothing about the level of the course in which it appears.

Math and Science	Computer Engineering Topics		Additional topics
	Core CPE Topics	Elective CPE Topics	(from engineering, mathematics, general studies, and other topics according to program objectives)
===== 1 year	===== 1 year	===== 0.5 year	===== 1.5 year

Figure 1. Organization of a computer engineering curriculum.

Assessing the time required to cover a unit

To give readers a sense of the time required to cover a particular unit, the CCCE Task Force, consistent with the Computer Science Report [5] and earlier curriculum reports [2], has chosen to express time in *hours*, corresponding to the in-class time required to present that material in a traditional lecture-oriented format. The time designated for each unit should be interpreted as the *minimum* amount of time necessary to enable a student to achieve the learning objectives for that unit. It is often appropriate to spend more time on a unit than the mandated minimum. Even though this metric has its roots in a classical lecture-oriented form, the CCCE Task Force does not seek to endorse the lecture format, believing there to be other styles—particularly given recent improvements in educational technology—that can be at least as effective. It should be noted that the hours specified do not include time spent outside of a class for homework, projects, and other academic activities.

The core hours, as specified in the Appendix, total 420 contact hours. Assuming a 15-week semester, a typical three credit hour course would have about 42 contact hours for presentation of material. The 420 core hours are thus roughly equivalent to ten three-credit hour courses or 30 semester credit hours. This number is approximately one quarter of the 128 credit hours as specified in a typical engineering program. This leaves ample room for the addition of laboratory courses, a capstone project, electives, and general studies, allowing an institution to customize their program. For example, as depicted in Figure 1, ABET currently requires one and one-half years of engineering topics and one year of mathematics and basic science. The discrete structures area and the probability and statistics area often apply as mathematics rather than engineering areas. The core hours listed for the other 16 knowledge areas in the Appendix would constitute approximately two-thirds of the required minimum engineering content.

Integration of engineering practice into the computer engineering curriculum

In developing the CCCE Report, it was essential that it go beyond the body of knowledge to discuss the integration of engineering practice into the computer engineering curriculum. Coverage of knowledge units alone is not enough. Given that computer engineers are, first and foremost, engineers, any curriculum in computer engineering must exhibit an engineering ethos. This should permeate all years of the curriculum and do so in a consistent manner. Such an approach has the effect of introducing students to engineering (and in particular computer

engineering), teaching them to think and function as engineers, and setting expectations for the future.

The basic engineering and personal skills necessary to enable the computer engineering graduate to apply this body of knowledge to real-world problems and situations are examined in Chapter 5 of the CCCE Report, addressing such aspects as engineering design, laboratory experience, and the role of engineering tools. Design throughout the curriculum and the culminating design experience are emphasized as critical elements of an engineering curriculum. In addition, such complementary skills as communication skills, teamwork, and lifelong learning are considered to be of equal importance to the computer engineer and are part of this chapter.

Professionalism

One aspect that makes computer engineers different from other computing professionals is their concentration on computer systems that include both hardware and software. Computer engineers design and implement computing systems that often affect the public. Computer engineers should hold a special sense of responsibility knowing that almost every element of their work can have a public consequence. Hence, computer engineers must consider the professional, societal, and ethical context in which they do their work. This context includes many issues such as intellectual property rights embodied by copyrights and patents, legal issues including business contracts and law practice, security and privacy issues as they apply to networks and databases, and liability issues as applied to hardware and software errors and economic issues as they apply to tradeoffs between product quality and profits. It also includes equity issues as they apply to technological access for all individuals. Computer engineers must be aware of the social context of their actions and be sensitive to the international implications of their activities. These issues are discussed in Chapter 6 of the CCCE Report.

Curriculum implementation issues

The creation of a complete degree program (an entire program of study) is far from straightforward. The body of knowledge does provide important input. However, many other influences contribute to the creation of a curriculum. Chapter 7 of the CCCE Report explores issues in the design and creation of a complete computer engineering degree program. These issues range from specifics such as packaging material from the BOK into introductory and advanced courses, designing courses exclusive to computer engineering vs. courses shared with other majors (ex. computer science and electrical engineering), selecting supporting mathematics and science courses, and integrating engineering practice into the curriculum, to more general considerations such as creating an overall style or ethos for a particular computer engineering degree program. This includes consideration of whether to design a broad curriculum or a program that focuses on one specific area of computer engineering, perhaps to serve the needs of local industries or to reflect interests and background of the faculty.

Several sample curriculum models, and syllabi of their component courses, are presented in Appendices B and C of the CCCE Report to illustrate different ways in which a computer engineering curriculum might be created from the body of knowledge and engineering practice aspects presented in this report. These curriculum models were designed from different perspectives to address the needs of different types of programs. These include a curriculum

created for a typical Electrical and Computer Engineering department, one from a smaller school where computer engineering might have evolved from the computer science program, an interdisciplinary curriculum derived from a computer science program and an electrical engineering program, and another that might represent a typical program in the United Kingdom.

Summary

The charter of the CCCE task force was to review the Joint ACM and IEEE/CS Computing Curricula 1991 and develop a report that specifically addresses computer engineering curricula that build on developments in computing technologies in the past decade and will sustain through the next decade. The final CCCE report presents an overview of the discipline of computer engineering, the computer engineering body of knowledge, and related engineering practice issues, and discusses issues affecting the implementation of a computer engineering curriculum. The bulk of the material in the report appears in three appendices: the body of knowledge for undergraduate computer engineering programs, descriptions for recommended courses that comprise the sample curricula, and sample curricula that might appear at different academic institutions.

The CCCE Task Force is hopeful that providing the body of knowledge, course descriptions, and sample curricula will help departments to create effective curricula or to improve the curricula they already have.

References

[1] Computing Curriculum 2001 website: http://www.computer.org/ education /cc2001.
[2] "Computing Curriculum 1991", Report of the ACM/IEEE-CS Joint Curriculum Task Force, 1991, IEEE Computer Society Press and ACM Press.
[3] "Computing Curriculum 2004 - Computing Curriculum: Computer Engineering", http://www.eng.auburn.edu/ece/CCCE
[4] "Computing Curricula Computer Engineering, Final Report", Joint Task Force of the IEEE Computer Society and Association for Computing Machinery on Computing Curricula Computer Engineering.
[5] "Computing Curriculum 2001 - Computing Curriculum: Computer Science", http://www.computer.org/education/ cc2001/final/ index.htm

Appendix
The Computer Engineering Body of Knowledge

Computer Engineering Knowledge Areas and Units	
CE-ALG Algorithms and Complexity [30 core hours] CE-ALG0 History and overview [1] CE-ALG1 Basic algorithmic analysis [4] CE-ALG2 Algorithmic strategies [8] CE-ALG3 Computing algorithms [12] CE-ALG4 Distributed algorithms [3] CE-ALG5 Algorithmic complexity [2] CE-ALG6 Basic computability theory	**CE-CAO Computer Architecture and Organization [63 core hours]** CE-CAO0 History and overview [1] CE-CAO1 Fundamentals of computer architecture [10] CE-CAO2 Computer arithmetic [3] CE-CAO3 Memory system organization and architecture [8] CE-CAO4 Interfacing and communication [10] CE-CAO5 Device subsystems [5] CE-CAO6 Processor systems design [10] CE-CAO7 Organization of the CPU [10] CE-CAO8 Performance [3] CE-CAO9 Distributed system models [3] CE-CAO10 Performance enhancements CE-CAO11 Crosscutting Issues
CE-CSE Computer Systems Engineering [18 core hours] CE-CSE0 History and overview [1] CE-CSE1 Life cycle [2] CE-CSE2 Requirements analysis and elicitation [2] CE-CSE3 Specification [2] CE-CSE4 Architectural design [3] CE-CSE5 Testing [2] CE-CSE6 Maintenance [2] CE-CSE7 Project management [2] CE-CSE8 Concurrent (hardware/software) design [2] CE-CSE9 Implementation CE-CSE10 Specialist systems CE-CSE11 System-level test and diagnosis CE-CSE12 Reliability and fault tolerance CE-CSE13 Error detecting and correcting codes	**CE-CSG Circuits and Signals [43 core hours]** CE-CSY0 History and overview [1] CE-CSY1 Electrical Quantities [3] CE-CSY2 Resistive Circuits and Networks [9] CE-CSY3 Reactive Circuits and Networks [12] CE-CSY4 Frequency Response [9] CE-CSY5 Sinusoïdal Analysis [6] CE-CSY6 Convolution [3] CE-CSY7 Fourier Analysis CE-CSY8 Filters CE-CSY9 Laplace Transforms
CE-DBS Database Systems [5 core hours] CE-DBS0 History and overview [1] CE-DBS1 Database systems [2] CE-DBS2 Data modeling [2] CE-DBS3 Relational databases CE-DBS4 Database query languages CE-DBS5 Relational database design CE-DBS6 Transaction processing CE-DBS7 Distributed databases CE-DBS8 Physical database design	**CE-DIG Digital Logic [57 core hours]** CE-DIG0 History and overview [1] CE-DIG1 Switching theory [6] CE-DIG2 Combinational logic circuits [4] CE-DIG3 Modular design of combinational circuits [6] CE-DIG4 Memory elements [3] CE-DIG5 Sequential logic circuits [10] CE-DIG6 Digital systems design [12] CE-DIG7 Modeling and simulation [5] CE-DIG8 Formal verification [5] CE-DIG9 Fault models and testing [5] CE-DIG10 Design for testability
CE-DSP Digital Signal Processing [17 core hours] CE-DSP0 History and overview [1] CE-DSP1 Theories and concepts [3] CE-DSP2 Digital spectra analysis [1] CE-DSP3 The discrete Fourier transform [7] CE-DSP4 Sampling [2] CE-DSP5 Transforms [2] CE-DSP6 Digital filters [1] CE-DSP7 Discrete time signals CE-DSP8 Window functions CE-DSP9 Convolution CE-DSP10 Speech processing	**CE-ELE Electronics [40 core hours]** CE-ELE0 History and overview [1] CE-ELE1 Electronic properties of materials [3] CE-ELE2 Diodes and diode circuits [5] CE-ELE3 MOS transistors and biasing [3] CE-ELE4 MOS logic families [7] CE-ELE5 Bipolar transistors and logic families [4] CE-ELE6 Design parameters and issues [4] CE-ELE7 Storage elements [3] CE-ELE8 Interfacing logic families and standard buses [3] CE-ELE9 Operational amplifiers [4] CE-ELE10 Circuit modeling and simulation [3] CE-ELE11 Data conversion circuits CE-ELE12 Electronic voltage and current sources CE-ELE13 Transistor amplifier design CE-ELE14 Power circuits CE-ELE15 Feedback in electronics CE-ELE16 Active filters CE-ELE17 Integrated circuit building blocks

CE-ESY Embedded Systems [20 core hours]
- CE-ESY0 History and overview [1]
- CE-ESY1 Embedded microcontrollers [6]
- CE-ESY2 Embedded programs [3]
- CE-ESY3 Real-time operating systems [3]
- CE-ESY4 Low-power computing [2]
- CE-ESY5 Reliable system design [2]
- CE-ESY6 Design methodologies [3]
- CE-ESY7 Tool support
- CE-ESY8 Embedded multiprocessors
- CE-ESY9 Networked embedded systems
- CE-ESY10 Interfacing and mixed-signal systems

CE-HCI Human-Computer Interaction [8 core hours]
- CE-HCI0 History and overview [1]
- CE-HCI1 Foundations of human-computer interaction [2]
- CE-HCI2 Graphical user interface [2]
- CE-HCI3 I/O technologies [1]
- CE-HCI4 Intelligent systems [2]
- CE-HCI5 Human-centered software evaluation
- CE-HCI6 Human-centered software development
- CE-HCI7 Interactive graphical user-interface design
- CE-HCI8 Graphical user-interface programming
- CE-HCI9 Graphics and visualization
- CE-HCI10 Multimedia systems

CE-NWK Computer Networks [21 core hours]
- CE-NWK0 History and overview [1]
- CE-NWK1 Communications network architecture [3]
- CE-NWK2 Communications network protocols [4]
- CE-NWK3 Local, wide area, and wireless networks [4]
- CE-NWK4 Client-server computing [3]
- CE-NWK5 Data security and integrity [4]
- CE-NWK6 Wireless and mobile computing [2]
- CE-NWK7 Performance evaluation
- CE-NWK8 Data communications
- CE-NWK9 Network management
- CE-NWK10 Compression and decompression

CE-OPS Operating Systems [20 core hours]
- CE-OPS0 History and overview [1]
- CE-OPS1 Design principles [5]
- CE-OPS2 Concurrency [6]
- CE-OPS3 Scheduling and dispatch [3]
- CE-OPS4 Memory management [5]
- CE-OPS5 Device management
- CE-OPS6 Security and protection
- CE-OPS7 File systems
- CE-OPS8 System performance evaluation

CE-PRF Programming Fundamentals [39 core hours]
- CE-PRF0 History and overview [1]
- CE-PRF1 Programming Paradigms [5]
- CE-PRF2 Programming constructs [7]
- CE-PRF3 Algorithms and problem-solving [8]
- CE-PRF4 Data structures [13]
- CE-PRF5 Recursion [5]
- CE-PRF6 Object-oriented programming
- CE-PRF7 Event-driven and concurrent programming
- CE-PRF8 Using APIs

CE-SPR Social and Professional Issues [16 core hours]
- CE-SPR0 History and overview [1]
- CE-SPR1 Public policy [2]
- CE-SPR2 Methods and tools of analysis [2]
- CE-SPR3 Professional and ethical responsibilities [2]
- CE-SPR4 Risks and liabilities [2]
- CE-SPR5 Intellectual property [2]
- CE-SPR6 Privacy and civil liberties [2]
- CE-SPR7 Computer crime [1]
- CE-SPR8 Economic issues in computing [2]
- CE-SPR9 Philosophical frameworks

CE-SWE Software Engineering [13 core hours]
- CE-SWE0 History and overview [1]
- CE-SWE1 Software processes [2]
- CE-SWE2 Software requirements and specifications [2]
- CE-SWE3 Software design [2]
- CE-SWE4 Software testing and validation [2]
- CE-SWE5 Software evolution [2]
- CE-SWE6 Software tools and environments [2]
- CE-SWE7 Language translation
- CE-SWE8 Software project management
- CE-SWE9 Software approaches and software fault tolerance

CE-VLS VLSI Design and Fabrication [10 core hours]
- CE-VLS0 History and overview [1]
- CE-VLS1 MOS Transistor Fundamentals [3]
- CE-VLS2 Processing and Layout
- CE-VLS3 Function of the Basic Inverter Structure [3]
- CE-VLS4 Circuit Characterization and Performance
- CE-VLS5 Combinational Logic Circuits
- CE-VLS6 Sequential Logic Circuits
- CE-VLS7 Alternative Circuit Structures/Low Power Design
- CE-VLS8 Semiconductor Memories and Array Structures [3]
- CE-VLS9 Chip Input/Output Circuits
- CE-VLS10 Semi custom Design Technologies
- CE-VLS11 ASIC Design Methodology

Mathematics Knowledge Areas and Units

CE-DSC Discrete Structures [33 core hours]
- CE-DSC0 History and overview [1]
- CE-DSC1 Functions, relations, and sets [6]
- CE-DSC2 Basic logic [10]
- CE-DSC3 Proof techniques [6]
- CE-DSC4 Basics of counting [4]
- CE-DSC5 Graphs and trees [4]
- CE-DSC6 Recursion [2]

CE-PRS Probability and Statistics [33 core hours]
- CE-PRS0 History and overview [1]
- CE-PRS1 Discrete probability [6]
- CE-PRS2 Continuous probability [6]
- CE-PRS3 Sampling distributions [4]
- CE-PRS4 Stochastic Processes [6]
- CE-PRS5 Sampling distributions [4]
- CE-PRS6 Estimation [4]
- CE-PRS7 Hypothesis tests [2]
- CE-PRS8 Correlation and regression