Computing Curricula

Computer Engineering

The Joint Task Force on Computing Curricula 2005
IEEE Computer Society
Association for Computing Machinery

Report

Ironman Draft

2004 June 8

CCCE Task Force Members

David Soldan (Chair)	Kansas State University
James Aylor	University of Virginia
Alan Clements	University of Teesside – England
Gerald Engel	University of Connecticut
Ron Hoelzeman	University of Pittsburgh
Esther A. Hughes	Virginia Commonwealth University
Joseph L.A. Hughes	Georgia Institute of Technology
John Impagliazzo	Hofstra University
Robert Klenke	Virginia Commonwealth University
Douglas A. Lyon	Fairfield University
Andrew McGettrick	University of Strathclyde – Scotland
Victor P. Nelson	Auburn University
Daniel J. Neebel	Loras College
Ivor Page	University of Texas – Dallas
Gregory D. Peterson	University of Tennessee – Knoxville
Robert Sloan	University of Illinois – Chicago
Pradip Srimani	Clemson University
Mitch Theys	University of Illinois – Chicago
Murali Varanasi	University of South Florida

Contents

Chapter 1

Introduction

In the fall of 1998, the Computer Society of the Institute for Electrical and Electronics Engineers (IEEE-CS) and the Association for Computing Machinery (ACM) established the Joint Task Force on "model Curricula for Computing" (or CC for short) to undertake a major review of curriculum guidelines for undergraduate programs in computing. The charter of the task force is as follows:

> To review the Joint ACM and IEEE/CS Computing Curricula 1991 and develop a revised and enhanced version that addresses developments in computing technologies in the past decade and will sustain through the next decade.

As indicated in the charter, the goal of the CC effort is to revise *Computing Curricula 1991* so that it incorporates the developments of the past decade. Computing has changed dramatically over that time in ways that have a profound effect on curriculum design and pedagogy. Moreover, the scope of what one calls *computing* has broadened to the point that it is difficult to define it as a single discipline. Previous curricula reports have attempted to merge such disciplines as computer science, computer engineering, and software engineering into a single report about computing education. While such an approach may have seemed reasonable in the past, there is no question that computing in the twenty-first century encompasses many vital disciplines with their own identities and pedagogical traditions.

Another part of the charter of this group includes supporting the community of professionals responsible for developing and teaching a range of courses throughout the global community. Providing an international perspective presents different challenges, but is an important ingredient given the global nature of computing related developments.

1.1 Overall Structure of the Computing Curricula Project

Due to the broadening scope of computing—and the feedback received on the initial draft — the CC initiative contains several reports. This report focuses specifically on computer engineering, referred to as "Computing Curricula: Computer Engineering" or simply CCCE. To encompass the different disciplines that are part of the overall scope of computing, professional organizations have created additional committees to undertake similar efforts in other areas. These areas include computer science ("Computing Curricula: Computer Science" or the CCCS report published in December 2001), information systems ("Computing Curricula: Information Systems" or the CCIS report published in 2002), software engineering ("Computing Curricula: Software Engineering" or the CCSE report currently under development), and information technology ("Computing Curricula: Information Technology" or the CCIT report currently under development).

As the individual reports unfold to completion, representatives from the five computing disciplines have produced an overview report that links them together. That overview report contains descriptions of the various computing disciplines along with an assessment of the commonalities and differences that exist among them. It also suggests the possibility of future curricular areas in computing. The structure of the series appears in Figure 1-1 as taken from the overview report. The area of information technology is the newest component of the computing curricula project.

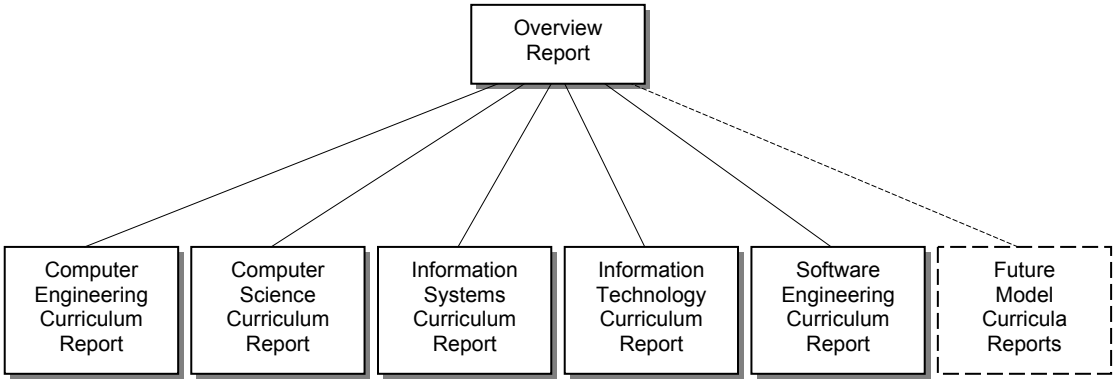

Figure 1.1: Computing curricula reports

1.2 Overview of the CCCE Process

In their charter, the main CC Steering Committee gave individual groups freedom to produce reports that best reflect the needs and requirements of their particular disciplines. However, the committee did request that groups address a certain minimal number of matters and, consequently, that they should include certain components in the individual reports. The minimal set includes:

- The body of knowledge (BOK) for the field; that is, the topics the field should cover,
- A set of courses that cover the body of knowledge in one or more ways,
- The core requirements for the discipline; that is, the requirements that would apply to all undergraduates, and
- The characteristics of graduates of degree programs

The Steering Committee viewed the set of requirements as minimal, as one of its goals was to avoid prescription. The experts must have the freedom to act as they see fit. Yet there must be some commonality across the different series of reports. The anticipation is that each report will exceed this minimal set in various ways.

In pursuing this charter, it is natural that the Computer Engineering Task Force be cognizant of what the Computer Science Task Force had already accomplished. The thrust of the Computer Engineering Task Force was to build on work already completed wherever possible.

Despite the considerable growth of computer engineering as a discipline, the literature in computer engineering curricular development is modest. There are a few contributions such as [Bennett 1986], [EAB 1986], and [Langdon, et. al. 1986]. The focus on the first three of these was not curricular development; they addressed issues such as resources and design processes. These issues are still important and appear elsewhere in this document.

To respond to the challenges of their charter, the Computer Engineering Task Force emerged from computer engineering interests from different countries. In addition, there was some overlap with the original Computer Science Task Force to ensure continuity. In discharging its duty, the Computer Engineering Task Force felt that it was vital to involve the wider community; indeed, several consultative activities occurred to confirm the view expressed in this volume. In addition, the task force used the world wide web [Aub] to allow any interested party the opportunity to provide comment and suggestion. The published report has benefited from this wide and important involvement

Developing the recommendations in this report is primarily the responsibility of the CCCE Task Force, the members of which appear at the beginning of this report. Given the scale of the CCCE project and the scope over which it extends, it was necessary to secure the involvement of many other people, representing a wide range of constituencies and areas of expertise.

1.3 Structure of the CCCE Report

This CCCE report addresses computer engineering programs. The main body of the report consists of eight chapters. Chapter 2 illustrates how computer engineering evolved as a discipline. It also highlights many of the characteristics expected of a computer engineering graduate, especially their service to the public, their design abilities, and their expected breadth of knowledge. It also suggests possible organizational structures, the responsibility of professional practices, and program assessment. Chapter 3 articulates the principles that have guided the work of the Computer Engineering Task Force and how these principles relate to *CC2001*. Chapters 4 and 5 present overviews of the computer engineering body of knowledge and curriculum recommendations. They also articulate learning objectives, the differences between core and elective knowledge units, the number of core hours in the program, the importance of design and laboratory experiences, and various skills needed to become an effective computer engineer. Chapter 6 highlights the importance of professionalism in the practice of computer engineering. Chapter 7 provides a discussion on the issues affecting the implementation of a computer engineering curriculum. These include the arrangement of courses within and external to the program and other implementation considerations. Chapter 8 suggests some challenges that need reviewing when creating or continuing computer engineering programs. This report provides two sets of references: those made within this report and a full set of references related to all computing curricula programs.

The bulk of the material in the report appears in two appendices. Appendix A addresses the body of knowledge in detail for undergraduate computer engineering programs. It includes all the computing knowledge areas, their associated knowledge units and related topics, student outcomes, and two related mathematics areas. Appendix B illustrates sample curricula and course descriptions, as they might appear at different academic institutions. The 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.

Chapter 2

Computer Engineering as a Discipline

This chapter presents some of the characteristics that distinguish computer engineering from other computing disciplines. It provides some background of the field and shows how it evolved over time. It also highlights some of the characteristics expected from its graduates, preparation for entering the curriculum, and student outcomes and assessment. The chapter also highlights the importance of graduates to have a proper sense of professionalism to ensure a proper perspective in the practice of computer engineering.

2.1 Background

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 been viewed as a combination of both computer science (CS) and electrical engineering (EE). It has 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 technical problems through the design of computing hardware, software, networks, and processes.

Historically, the field of computer engineering has been widely viewed as "designing computers." In reality, the design of computers themselves has been the province of relatively few highly skilled engineers whose goal was to push forward the limits of computer and microelectronics technology. The successful miniaturization of silicon devices and their increased reliability as system building blocks has created an environment in which computers have replaced the more conventional electronic devices. These applications manifest themselves in the proliferation of mobile telephones, personal digital assistants, location-aware devices, digital cameras, and similar products. It also reveals itself in the myriad of applications involving embedded systems, namely those computing systems that appear in applications such as automobiles, large-scale electronic devices, and major appliances.

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, 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. Applications include consumer electronics (CD and DVD players, televisions, stereos, microwaves, gaming devices) and advanced microprocessors, peripheral equipment, systems for portable, desktop and client/server computing, and communications devices (cellular phones, pagers, personal digital assistants). It also includes distributed computing environments (local and wide area networks, wireless networks, internets, intranets), and embedded computer systems (such as aircraft, spacecraft, and automobile control systems in which computers are embedded to perform various functions). A wide array of complex technological systems, such as power generation and distribution systems and modern processing and manufacturing plants, rely on computer systems developed and designed by computer engineers.

Technological advances and innovation continue to drive computer engineering. There is now a convergence of several established technologies (such as television, computer, and networking technologies) resulting in widespread and ready access to information on an enormous scale. This 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.

2.2 Evolution of the Field

As noted previously, computer engineering evolved from the disciplines of electrical engineering and computer science. Initial curricular efforts in computer engineering commonly occurred as a specialization within EE programs, extending digital logic design to the creation of small-scale digital systems and, eventually, the design of microprocessors and computer systems.

In the United States, the first computer engineering program accredited by ABET (formerly the Accreditation Board for Engineering and Technology) was at Case Western Reserve University in 1971. As of 2003 December, ABET has accredited over 150 computer engineering or similarly named programs. Table 2-1 summarizes the growth in programs by title and year of initial ABET accreditation (or change of program name). As a point of comparison, there are approximately 300 accredited electrical engineering programs.

Table 2-1
Summary of ABET-accredited computer engineering programs in the U.S. - as of 2003 December

	Year of Initial Accreditation				
Program Name	Before 1980	1980 to 1989	1990 to 1999	2000 to 2003	Totals
Computer Engineering	10	32	44	43	129
Computer Systems Engineering	2	2	0	1	5
Electrical and Computer Engineering (includes programs previously named EE)	2	4	0	5	11
Computer Science and Engineering	2	6	1	1	10
Other titles	0	2	1	1	4
Totals	16	46	46	51	159

One would expect that the growth trend in computer engineering will increase as computing and electronic technologies become more complex. The evolution may take many forms, including (a) an expanded content from computer science, (b) collaboration with the emerging software engineering discipline on application-focused projects and embedded systems with a greater emphasis on design and analysis tools to manage complexity, or (c) re-integration with electrical engineering, as computer-based systems become dominant in areas such as control systems and telecommunications.

2.3 Characteristics of Computer Engineering Graduates

With the ubiquity of computers and computer-based systems in the world today, computer engineers must be versatile in the knowledge drawn from standard topics in computer science and electrical engineering as well as the foundations in mathematics and sciences. Because of the rapid pace of change in the computing field, computer engineers must be life-long learners to maintain their knowledge and skills within their chosen discipline.

2.3.1 Distinctions

An important distinction should be made between computer engineers, electrical engineers, other computer professionals, and engineering technologists. While such distinctions are sometimes ambiguous, computer engineers generally should satisfy the following three characteristics.

1. Possess the ability to design computer systems that include both hardware and software to solve novel engineering problems, subject to trade-offs involving a set of competing goals and constraints. In this context, “design” refers to a level of ability beyond “assembling” or “configuring” systems.
2. Have a breadth of knowledge in mathematics and engineering sciences, associated with the broader scope of engineering and beyond that narrowly required for the field.
3. Acquire and maintain a preparation for professional practice in engineering.

In contrast, electrical engineers concern themselves mostly with the physical aspects of electronics including circuits, signal analysis, and microelectronic devices. Computer scientists concern themselves primarily with the theoretical and algorithmic aspects of computing with a focus on the theoretical underpinnings of computing. Software engineers have a focus on the principles underlying the development and maintenance of correct (large-scale) software throughout its lifecycle. Information systems specialists encompass the acquisition, deployment, and management of information resources for use in organizational processes. Information technology specialists would focus on meeting the needs of users within an organizational and societal context through the selection, creation, application, integration, and administration of computing technologies. Computer engineering technologists concern themselves with making computer-based products work properly and in the maintenance of those products.

2.3.2 Professionalism

The public has entrusted in engineers a level of responsibility because the systems they design (whether x-ray machines, air traffic control systems, or nuclear power plants) affect the public directly and indirectly. Therefore, it is incumbent upon computer engineers to exercise the utmost conscientiousness when designing and implementing computing systems. As such, graduates should have an understanding of the responsibilities associated with engineering practice, including the professional, societal, and ethical context in which they do their work. Such responsibilities often involve complicated trade-offs involving fiscal and social contexts. This social context encompasses a range of legal and economic issues such as intellectual property rights, security and privacy issues, liability, technological access, and global implications and uses of technologies.

Professionalism and ethics are critical elements, since the focus of engineering on design and development makes social context paramount to studies in the field. Computer engineering students must learn to integrate theory, professional practice, and social constructs in their engineering careers. It is incumbent upon all computer engineers to uphold the tenets of their profession and to adhere to the codes of professional practice. The public expects engineers to follow prescribed rules of professional practice and to not engage in activities that would tarnish their image or that of their practicing colleagues. Because of the importance of this topic, Chapter 6 of this report is devoted to an expanded discussion on professional practice and responsibilities.

2.3.3 Ability to Design

Engineering draws heavily on the ability to design. The International Technology Education Association (ITEA) defines engineering design as “The systematic and creative application of scientific and mathematical principles to practical ends such as the design, manufacture, and operation of efficient and economical structures, machines, processes, and systems.” [ITEA] Other definitions are possible such as the creative ability required for the development of better devices, systems, processes, and new products. Many reasons prompt new designs such as seeking to exploit new developments in related technologies or to develop improvements on existing products (e.g. making products less expensive, safer, more flexible, or lighter in weight). Identifying deficiencies or weaknesses in existing products is another motivation for engineering design. Of course, novel ideas are especially important.

Design is fundamental to all engineering. For the computer engineer, design relates to software and hardware components of modern computing systems and computer-controlled equipment. Computer engineers apply the theories and principles of science and mathematics to design hardware, software, networks, and processes and to solve technical problems. Continuing advances in computers and digital systems have created opportunities for professionals capable of applying these developments to a broad range of applications in engineering. Fundamentally, it is about making well-considered *choices* or *trade-offs*, subject to given constraints. These relate to such matters as structure and organization, techniques, technologies, methodologies, interfaces, as well as the selection of components. The outcome needs to exhibit desirable properties and these tend to relate to simplicity and elegance. Chapter 5 presents a more detailed discussion of design and related laboratory experiences.

2.3.4 Breadth of Knowledge

Because of the breadth of the computer-engineering field, curricular content may vary widely among programs, or even among students in the same program. Computer-related coursework typically comes from computer organization and architecture, algorithms, programming, databases, networks, software engineering, and

communications. Electrical engineering related coursework typically comes from circuits, digital logic, microelectronics, signal processing, electromagnetics, and integrated circuit design. Foundational topics typically include basic sciences, mathematics for both discrete and continuous domains, and applications of probability and statistics.

At one extreme, a degree program in computer engineering might provide opportunities for its students to study a wide range of topics spanning the entire field. At another extreme, there may be programs that focus on one specific aspect of computer engineering and cover it in great depth. The graduates from such programs will typically tend to seek opportunities in the specialist area they studied, whether it is multimedia systems development, computer design, network design, safety-critical systems, pervasive computing, or whatever other specialties emerge and become important. One common measure for differentiating among computer engineering programs is the relative amount of emphasis placed on topics that are commonly associated with either electrical engineering or computer science programs.

Despite differences in emphasis and content, there are certain common elements that one should expect of any computer engineering program. The Body of Knowledge, described in Chapter 4, identifies topical areas that one may reasonably expect in all programs, as opposed to those that are often included in some programs or those that one might consider elective or specialized topics. From a higher-level perspective, however, one can reasonably expect several characteristics of all computer engineering graduates. These include:

- *System Level Perspective* – Graduates must appreciate the concept of a computer system, the design of the hardware and software for that system, and the processes involved in constructing or analyzing it. They must have an understanding of its operation that goes to a greater depth than a mere external appreciation of what the system does or the way(s) in which one uses it.
- *Depth and Breadth* – Graduates should have familiarity with topics across the breadth of the discipline, with advanced knowledge in one or more areas.
- *Design Experiences* – Graduates should have completed a sequence of design experiences, encompassing hardware and software elements, building on prior work, and including at least one major project.
- *Use of Tools* – Graduates should be capable of utilizing a variety of computer-based and laboratory tools for the analysis and design of computer systems, including both hardware and software elements.
- *Professional Practice* – Graduates should understand the societal context in which engineering is practiced, as well as the effects of engineering projects on society.
- *Communication Skills* – Graduates should be able to communicate their work in appropriate formats (written, oral, graphical) and to critically evaluate materials presented by others in those formats.

2.4 Organizational Considerations

The administration of computer engineering programs falls within a variety of organizational structures. Currently, computer engineering programs are rarely organized as separate academic departments. They often appear in colleges or schools of engineering, either within an electrical engineering department, within a combined engineering department, or within an electrical and computer engineering department. In such cases, the expectation is a strong emphasis on circuits and electronic components. Computer engineering programs also appear in areas such as computer science departments, colleges of arts and sciences, schools or divisions of information technology, or co-sponsored by multiple entities. In these cases, the programs often relate more to the issues of theory, abstraction, and organization rather than those of a more applied nature.

As noted in Table 2-1, the most common degree title for these programs is “Computer Engineering.” Other titles may reflect program specializations, organizational structures, historical constraints, or other factors. The principles presented in this report apply to all computer engineering programs regardless of their organizational structure or official degree title.

2.5 Preparation for Professional Practice

Unlike professions such as law and medicine, engineering generally does not require an advanced degree for employment in the field. Thus, undergraduate programs in computer engineering must include not only basic knowledge within the field, but the ability to apply it to the solution of realistic projects. This preparation encompasses several areas.

Section 2.3.2 defined the professionalism and ethics that are fundamental characteristics of a computer engineering graduate. Preparation for professional practice requires graduates to have an understanding of their responsibilities associated with engineering practice, as well as an ability to apply these principles to specific situations. Professionalism should be a constant theme that pervades the entire curriculum. In particular, the social context of engineering should be integrated into the teaching of engineering design, including the use of best practices and trade-offs among technical, fiscal, and social requirements.

In addition to professionalism, appropriate preparation encompasses both technical (design ability, laboratory experiences, use of engineering tools) and non-technical (teamwork, communication) elements. Chapter 5 of this report provides a detailed discussion on the integration of these issues into the curriculum.

2.6 Program Evaluation and Accreditation

Processes for program evaluation must accommodate the variations among computer engineering programs. Such evaluation is critical to ensure that graduates have the proper preparation and that programs are evolving to meet the emerging requirements of the field. Often, professional societies and governments look toward an external assessment of programs to ensure that graduates achieve minimally what professional organizations expect of them.

Within the United States, ABET accreditation is widely recognized and accepted. The current engineering criteria [ABET, 2004] are intended to ensure that all accredited programs satisfy a minimum set of criteria common to all engineering disciplines and criteria specific to each discipline. A key element of this process is a requirement that each program engage in an ongoing process of self-assessment and continuous improvement. Programs should demonstrate that all graduates achieve a set of program outcomes based on the program's educational objectives. The ABET criteria are broadly defined. They leave the interpretation of what constitutes the appropriate knowledge for a given discipline to the professional societies affiliated with that discipline. We anticipate that this report will provide guidance to accrediting agencies on the appropriate technical content of computer engineering programs.

In the United Kingdom, benchmarking of degrees has developed in recent years and each institution is required to demonstrate that their degrees meet the requisite benchmark standards for that discipline. One example of these benchmark standards is [UKQAA2000]. This benchmarking defines both threshold (minimal) and modal (average) expectations with respect to demonstrated student knowledge, skills, and judgment. An example of a [threshold/modal] criterion is the following:

> Graduates will be able to produce work involving problem identification, the analysis, the design and the development of a system with appropriate documentation. The work will show [some / a range of] problem solving and evaluation skills drawing on [some/] supporting evidence, and demonstrate a [requisite/good] understanding of the need for quality.

The Engineering Council UK has overall responsibility for the accreditation of engineering degrees within the United Kingdom. Its basic responsibilities include setting standards (of competence and commitment) for the accreditation of engineering degrees and approving *nominating bodies* that carry out detailed accreditation on its behalf. In general, the British Computer Society (BCS) carries out accreditation of computing degree programs and the Institute of Electrical Engineers (IEE) carries out the accreditation of electronic and electrical engineering degree programs. Degree programs in computer engineering could be accredited by either society, though perhaps more often by IEE. However, joint accreditation by both societies is common. Links with professional engineers in other countries exist through the mechanisms of the Washington Accord, the Sydney Accord, the Dublin Accord, FEANI, and the International Register for Engineers.

The accreditation process in engineering in the UK dates back to around 1978. Over the years, there have been various formulations of the rules and criteria for accredited degrees. But in broad terms, the expectation is to have degrees that

- Encourage and foster an engineering ethos. This includes attention to such matters as design and a problem solving approach. Invariably, such degrees must include (normally in their final year) an individual project in which students have to demonstrate their ability to tackle and solve a substantial problem of a technical nature.
- Present a challenge and address appropriate underpinnings and theoretical considerations.
- Address the environmental concerns as well as the professional, legal, and ethical concerns associated with engineering including preparation for life-long learning.
- Have strong and effective input from industry into curriculum design and perspective including relevance to industry; manifestations of industry involvement include the existence of industrial scholarships or prizes as well as a willingness to host external activities such as internships.

Although there are some unique aspects to each accreditation agency, there also are many common elements:

1. The accreditation review process involves a visit to the institution by a panel of experts who meet staff and students and produce a report with accompanying recommendations about accreditation status.
2. Typical periods of accreditation range from zero to six years; the length of period generally reflects the confidence that the visiting panel has in the program.
3. A major goal of the process is one of support and development, with every attempt made to encourage and foster good practice.
4. Graduation with an accredited degree plus an appropriate period (typically about two years) of relevant industrial experience can lead to the award in the UK of the accolade of Chartered Engineer. Professionals generally regard designation as a well-qualified engineer; other routes to Chartered Engineer also exist. In the United States, graduation from an accredited engineering program is the initial step towards licensure as a professional engineer.

In general, institutions tend to use accreditation as a vehicle to provide evidence of quality that they can use in marketing activities; most institutions offering engineering degrees will have some form of recognition in accreditation terms. Currently, some jobs demand accredited degree status or professional licensure, although this requirement is not as widespread in computing-related fields as in some other engineering fields.

While accreditation and benchmarking standards typically refer to the minimum or average graduate, the expectation is that computer engineering programs also will provide opportunities for the best students to achieve their full potential. Such students will be creative and innovative in their application of the principles covered in the curriculum; they will be able to contribute significantly to the analysis, design, and development of complex systems; and they will be able to exercise critical evaluation and review of both their own work and the work of others.

Chapter 3

Principles

Computing is a growing and important area of endeavor. The Computer Engineering Task Force established a set of principles to guide its work that reflects in part those that appeared in the Computer Science Report. They appear here with appropriate rewording and modification to reflect better the tenets expected from a computer engineering program. The presentation here is not in order of priority.

1. *Computer engineering is a broad and developing field.* The original CC Steering Committee had taken the view that a single report, covering primarily computer science, could not address the full range of issues that colleges and universities have to consider as they seek to address their computing curricula, and that a different task force should develop a separate report addressing computer engineering.

2. *Computer engineering is a distinct discipline* with its own body of knowledge, its own ethos, and its own practices. That discipline embodies the science and the technology of specification, design, construction, implementation, and maintenance of the hardware and software components of modern computer systems and computer-controlled equipment.

3. *Computer engineering draws its foundations from a wide variety of other disciplines.* Computer engineering education is solidly grounded in the theories and principles of computing, mathematics, and engineering, and it applies these theoretical principles to design hardware, software, networks and computerized equipment and instruments to solve technical problems in diverse application areas.

4. *The rapid evolution of computer engineering requires an ongoing review of the corresponding curriculum.* Given the pace of change in the discipline, the professional associations in this discipline must establish an ongoing review process that allows the timely update of the individual components of the curriculum recommendations.

5. *Development of a computer engineering curriculum must be sensitive to changes in technology, new developments in pedagogy, and the importance of lifelong learning.* In a field that evolves as rapidly as computer engineering, educational institutions must adopt explicit strategies for responding to change. Computer engineering education must seek to prepare students for lifelong learning that will enable them to move beyond today's technology to meet the challenges of the future.

6. *The Computer Engineering Task Force should seek to identify the fundamental skills and knowledge that all computer engineering graduates must possess.* Computer engineering is a broadly based discipline. The final report must seek to identify the common concepts and skills of the discipline.

7. *The required core of the body of knowledge should be as small as reasonably possible.* The Task Force should make every effort to keep that core to a minimum to allow flexibility, customization, and choice in other parts of the curriculum to enable creation of individualized programs.

8. *Computer engineering must include appropriate and necessary design and laboratory experiences.* A computer engineering program should include "hands-on" experience in designing, building, and testing both hardware and software systems.

9. *The computer engineering core acknowledges that engineering curricula are often subject to accreditation, licensure, or governmental constraints.* This computer engineering report recognizes existing external constraints and is intended to provide guidance for their evolution.

10. *The computer engineering curriculum must include professional practice as an integral component.* These practices encompass a wide range of activities including management, ethics and values, written and oral communication, working as part of a team, and remaining current in a rapidly changing discipline.

11. *The computer engineering report must include discussions of strategies and tactics for implementation along with high-level recommendations.* Although it is important for computing curricula to articulate a broad vision of computing education, the success of any curriculum depends heavily on implementation details. To accomplish this, the report should provide sample curricula models.

12. *The development of the final report must contain a broad base.* To be successful, the process of creating the computer engineering recommendations must include participation from many different constituencies including industry, government, and the full range of higher educational institutions involved in computer engineering education.

13. *The computer engineering final report must strive to be international in scope.* Despite the fact that curricular requirements differ from country to country, this report must be useful to computing educators throughout the world. Although educational practice in the United States may influence curriculum, the report makes every effort to ensure that the curriculum recommendations are sensitive to national and cultural differences so that they will be widely applicable throughout the world.

Chapter 4

Overview of the Computer Engineering Body of Knowledge

Developing any curriculum for undergraduate study in computer engineering should reflect the current needs of computer engineering students. The curriculum should also reflect current educational practice and suggest improvements where necessary. The discussion that follows attempts to accomplish this in preparing a body of knowledge commensurate with producing competent computer engineering graduates.

4.1 The Body of Knowledge

The Computer Engineering Task Force has sought to assemble a modern curriculum by first defining the primary disciplines that make up the body of knowledge for computer engineering. Some of these discipline areas contain material that should be part of *all* computer engineering curricula. These are the 18 knowledge areas, including two covering related mathematics topics, listed in Table 4.1. Other areas contain material that might, or might not, be part of such curricula, depending on the specific educational objectives of a program. Some of these are listed in Chapter 7, but are not described in detail in this report.

Table 4.1
CCCE Discipline Areas Containing Core Material

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

After defining the above areas, each task force member designed and reviewed initial drafts defining the body of knowledge for one or more areas. In some cases, new members joined the task force to cover areas of expertise outside of those originally represented. Subsequently, a second task force member reviewed and revised each initial draft. After each revision, the entire task force reviewed the resulting draft for comment. At the completion of this process, the entire task force met as a group to review the draft body of knowledge, with follow-up modifications made as appropriate.

The task force released the resulting document for public review. It solicited reviews at a number of meetings, conferences, and other sources. The task force held an NSF-sponsored workshop in November 2002 in conjunction with the Frontiers in Education Conference [FIE'02] in Boston. Reviewers from academia and industry participated in the workshop and provided comments on the preliminary versions of the body of knowledge. Members from the task force presented and discussed the body of knowledge at a variety of conferences through panel discussions and poster sessions. Presentations to date appear in Table 4.2. The entire CCCE project has been available at [Aub] since 2002.

Table 4.2
CCCE Presentations

Date	Conference or meeting	Type
2002 June 16-19	American Society for Engineering Education (ASEE 2002) – Montreal [ASEE'02]	Panel
2002 November 6-9	Frontiers in Education – Boston [FIE'02]	Panel
2003 February 19-23	SIGCSE Technical Symposium – Reno [SIGCSE'03]	Panel
2003 March	Electrical and Computer Engineering Department Heads Association - Hawaii	Panel
2003 June 22-25	American Society for Engineering Education (ASEE 2002) – Nashville [ASEE'03]	Panel
2003 June 29 – July 2	Innovation and Technology in Computer Science Education – Greece [ITiCSE'03]	Poster
2003 November 5-8	Frontiers in Education - Denver [FIE'03]	Panel &Paper
2004 March 3-7	SIGCSE Technical Symposium – Norfolk [SIGCSE'04]	Panel
2004 June 21-24	MultiConference in Computer Sci. & Computer Engineering – Las Vegas [MCSCE'04]	Paper
2004 June 20-23	American Society for Engineering Education (ASEE 2002) – Salt Lake City [ASEE'04]	Paper
2004 June 29-30	Innovation and Technology in Computer Science Education – England [ITiCSE'04]	Poster
2004 October 20-23	Frontiers in Education – Savannah [FIE'04]	Panel & Paper

4.2 Structure of the Body of Knowledge

The body of knowledge has a hierarchical organization comprising three levels described as follows.

- The highest level of the hierarchy is the *knowledge area*, which represents a particular disciplinary sub-field. A three-letter abbreviated tag identifies each area, such as CE-DIG for "Digital Logic" and CE-CAO for "Computer Architecture and Organization."
- Each knowledge area is broken down into smaller divisions called *knowledge units*, which represent individual thematic modules within an area. A numeric suffix added to the area name identifies each knowledge unit. For example, CE-CAO3 is a knowledge unit on "Memory System Organization and Architecture" within the CE-CAO knowledge area.
- A set of *topics*, which are the lowest level of the hierarchy, further subdivides each knowledge unit. A group of learning outcomes addresses the related technical skills associated with each knowledge unit. Section 4.3 expands the discussion on learning outcomes.

To differentiate knowledge areas and knowledge units in computer engineering from those that may have the same or similar names in the other four curriculum areas associated with this computing curriculum project, the prefix "CE-" accompanies all knowledge areas and units in computer engineering. Reflecting the examples above, therefore, tags such as CE-DIG for knowledge areas and CE-CAO3 for knowledge units appear throughout the report.

4.3 Learning Outcomes

To capture the various skills associated with obtaining knowledge, this report uses the phrase *learning outcomes* as a component of each knowledge unit. The emphasis on *learning* is important. The concept of *learning outcomes* is a mechanism for describing not just knowledge and relevant practical skills, but also personal and transferable skills. Outcomes can be associated with a knowledge unit, a class, a course, or even a degree program. Teachers can use them to convey different aspects of the ethos of a course or area of study.

Any specification of a course will include both knowledge and associated learning outcomes. In designing courses, some designers start with knowledge while others start with the learning outcomes. In reality, a combination of the two approaches appears most appropriate. In addition, a certain duality exists between the elements of knowledge and the related learning outcomes or objectives. Different people will place different levels of emphasis on each. For this document, the view is that they are complementary.

Since learning outcomes imply assessment and since assessment guides learning, teachers should exercise considerable care in selecting and formulating these. Excessive numbers of very detailed learning outcomes can lead to bureaucracy and tedium, which is highly undesirable. The existence of these outcomes must not inhibit course development; it should enhance that activity.

Learning outcomes are part of knowledge units and can be part of *modules*, which constitute the formal units of assessment. The number of learning outcomes per knowledge unit or module should be a small number—at most four or five. The learning outcomes for a module will naturally build on the knowledge units and the associated practical skills. They tend to be of the form:

> Demonstrate the acquisition of competence; that is, show the ability to apply knowledge and practical skills to solve a problem.

Of course, the ways of demonstrating skills can be many and varied; in particular, they can involve a range of communication and other skills. In this way, imaginative approaches to assessment can lead to the assessment of a range of skills in a well-conceived assignment.

4.4 Core and Elective Knowledge Units

As computer engineering evolves, the number of topics required in the undergraduate curriculum is growing. Over the last decade, computer engineering has expanded to such an extent that it is no longer possible to add new topics without taking others away. One of the goals in proposing curricular recommendations is to keep the required component of the body of knowledge as small as possible.

To implement this principle, the Computer Engineering Task Force has defined a minimal *core* comprising those knowledge units for which there is broad consensus that the corresponding material is essential to anyone obtaining an undergraduate degree in computer engineering. The core is considered essential, independent of the specific program degree title or organizational structure. Knowledge units presented as part of an undergraduate program, but which fall outside the core, are *elective* to the curriculum. Based on program goals, an institution may deem many elective units and areas as essential and require them for its program.

In discussing the recommendations during their development, the Task Force has found that it helps to emphasize the following important points.

- *The core is not a complete curriculum.*
  The intention of the core is minimal and it does *not* constitute a complete undergraduate curriculum. Every undergraduate program must include additional elective knowledge units from the body of knowledge. This report does not define what those units should be; that decision is the choice of each institution. A complete curriculum must also contain supporting areas covered through courses in mathematics, natural sciences, business, humanities, and/or social sciences. Chapter 7 presents some detail in this area.

- *Core units are not necessarily limited to a set of introductory courses taken early in the undergraduate curriculum.*
  Many of the knowledge units defined as core are indeed introductory. However, some core knowledge can appear only after students have developed significant background in the field. For example, the Task Force believes that all students must develop a significant application at some point during their undergraduate program. The material that is essential to successful management of projects at this scale is obviously part of the core, since it is required of all students. At the same time, the project course experience is very likely to

come toward the end of a student's undergraduate program. Similarly, introductory courses may include elective knowledge units together with the coverage of core material. From a practical point of view, the designation *core* simply means *required* and says nothing about the level of the course in which it appears.

4.5 Knowledge Units and Time Required for Coverage

To provide readers a sense of the time required to cover a particular unit, this report defines a metric that establishes a standard of measurement. Choosing such a metric has proven difficult, because no standard measure has global recognition. For consistency with the computer science report and earlier curriculum reports, the Task Force has chosen to express time in *hours*, corresponding to the in-class time required to present that material in a traditional lecture-oriented format. To dispel any potential confusion, however, it is important to underscore the following observations about the use of lecture hours as a measure.

- *The Task Force does not seek to endorse the lecture format.* Even though this report refers to a metric with its roots in a classical lecture-oriented form, the Task Force believes there are other styles - particularly given recent improvements in educational technology - that can be at least as effective. For some of these styles, the notion of hours may be difficult to apply. Even so, the time specifications should at least serve as a comparative measure, in the sense that a five-hour unit will presumably take roughly five times as much time to cover as a one-hour unit, independent of the teaching style.

- *The hours specified do not include time spent outside of a class.* The time assigned to a unit does not include the instructor's preparation time or the time students spend outside of class. As a general guideline, the amount of out-of-class work for a student is approximately three times the in-class time. Thus, a unit that is listed as requiring three hours will typically entail a total of twelve hours (three in-class hours and nine outside hours) of student effort.

- *The hours listed for a unit represent a minimum level of coverage.* One should interpret the time measurements assigned to each knowledge unit as the minimum amount of time necessary to enable a student to perform the learning objectives for that unit. It may be appropriate to spend more time on a knowledge unit than the mandated minimum.

4.6 Core Hours and a Complete Program

The knowledge units designated as core constitute only a fraction (approximately 30%) of the total body of knowledge. Different computer engineering programs can have different program objectives and as a result, will have different emphases. The remainder of a specific program at an institution usually will require specific additional knowledge units that complement the core areas, as well as elective hours chosen by individual students. Thus, each local program should seek to encompass that portion of the body of knowledge relevant to its program goals.

A summary of the body of knowledge—showing the areas, units, which units are core, and the minimum time required for each—appears in Table 4-3. It consists of 18 knowledge areas; 16 relate directly to computer engineering and 2 relate to mathematics (discrete structures, probability and statistics). The Computer Engineering Task Force has singled out these two mathematics areas as core because some programs may not consider them essential to computer engineering, as they would consider calculus. The details of the body of knowledge for computer engineering appear in Appendix A.

Table 4-3
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-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 Specialized systems CE-CSE11 Reliability and fault tolerance	**CE-CSG Circuits and Signals** [43 core hours] CE-CSG0 History and overview [1] CE-CSG1 Electrical Quantities [3] CE-CSG2 Resistive Circuits and Networks [9] CE-CSG3 Reactive Circuits and Networks [12] CE-CSG4 Frequency Response [9] CE-CSG5 Sinusoidal Analysis [6] CE-CSG6 Convolution [3] CE-CSG7 Fourier Analysis CE-CSG8 Filters CE-CSG9 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 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 Audio processing CE-DSP11 Image 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 Amplifier design CE-ELE14 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 and wide area 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 fault tolerance	**CE-VLS VLSI Design and Fabrication** [10 core hours] CE-VLS0 History and overview [1] CE-VLS1 Electronic properties of materials [2] CE-VLS2 Function of the basic inverter structure [3] CE-VLS3 Combinational logic structures [1] CE-VLS4 Sequential logic structures [1] CE-VLS5 Semiconductor memories and array structures [2] CE-VLS6 Chip input/output circuits CE-VLS7 Processing and layout CE-VLS8 Circuit characterization and performance CE-VLS9 Alternative circuit structures/low power design 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 Expectation [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

The core hours as specified in Table 4-3 total 420 hours of computer engineering and 66 hours of mathematics. Recall that an hour refers to a lecture hour and not a credit hour. Assuming a typical 15-week semester, a typical three-credit-hour course would have about 42 lecture hours for presentation of material. That is, approximately 14 lecture hours are equivalent to 1 semester credit hour. The 420 core computer engineering hours are thus roughly equivalent to ten three-credit-hour courses or 30 semester credit hours. The 30 semester credit hours is approximately one quarter of the 128 credit hours included in a typical four-year engineering program. The 420 core hours leave ample room for the addition of laboratory courses, a culminating design project, and electives that allow an institution to customize their program.

In the United States, for example, ABET accreditation criteria currently requires one and one-half years (approximately 48 semester hours) of engineering topics; it also requires one year (32 semester hours) of mathematics and basic science. The 48 semester hours are equivalent to 672 contact hours. Therefore, the 420 core hours listed in Table 4.3 would constitute approximately two-thirds of the required minimum engineering content.

Programs often categorize the discrete structures area and the probability and statistics area as mathematics rather than engineering or computing areas.

Figure 4.1 illustrates a four-year model program. It includes 1.0 year of mathematics and science, 1.0 year of computer engineering core, 0.5 year of computer engineering electives, 0.5 year of additional engineering studies, and 1.0 year of general studies. The model is adaptable to any worldwide system of study. In those countries where general studies precede university studies, a three-year model may be created, as shown in Figure 4.2, by removing the year of general studies and introductory mathematics and science. Appendix B includes examples of both four-year and three-year curricula.

Math and Science	**Computer Engineering Topics**		**Additional Topics**
	Core CPE Topics	Elective CPE Topics	(from engineering, mathematics, general studies, and other topics based on program objectives)
1 year	1 year	0.5 years	1.5 years

Figure 4.1. Organization of a four-year computer engineering curriculum.

Math and Science	**Computer Engineering Topics**		**Additional Topics**
	Core CPE Topics	Elective CPE Topics	(from engineering, mathematics, and other topics based on program objectives)
1 year	1 year	0.5 years	1.5 years

Figure 4.2. Organization of a three-year computer engineering curriculum.

Chapter 5

Integration of Engineering Practice into the Computer Engineering Curriculum

By its very nature, any curriculum in computer engineering should reflect an engineering ethos that permeates all years of the curriculum 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. Preparation for professional practice is essential since engineering, unlike such professions as law and medicine, generally does not require an advanced degree for employment in the field.

The role of this chapter is to go beyond the body of knowledge introduced in Chapter 4 and to examine the basic skills necessary to enable the computer engineering graduate to apply this body of knowledge to real-world problems and situations. Chapter 6 will then address the important matter of professionalism, and Chapter 7 will consider overall curriculum design, along with introducing sample curriculum implementations given in Appendix B.

5.1 The Nature of Computer Engineering

An important initial aspect of the engineering ethos relates to acquiring the background necessary to understand and to reason about engineering concepts and artifacts. This background stems from fundamental ideas in areas such as computing, electronics, mathematics and physics and students need to acquire familiarity and facility with these concepts. An important role of the body of knowledge for computer engineering is to expose and develop these fundamental notions. In many ways, the core of the body of knowledge reflects a careful set of decisions about selection of material that fulfils this role.

This basic material then provides underpinning for additional material whose ultimate expression is the building of better as well as novel computing systems. A blend of theory and practice, with theory guiding practice, appears to be the best approach to the discipline. The curriculum should accompany this blend with attention to a set of professional, ethical, and legal concerns that guide the activities and attitudes of the well-educated computer engineer. The curriculum should also foster familiarity with a considerable range of diverse applications.

5.2 Design in the Curriculum

In Chapter 2 of this report, a brief discussion on the characteristics of a computer engineer included the ability to design and provided a definition of engineering design.

5.2.1 Design Throughout the Curriculum

The principles of engineering design must pervade the entire computer engineering curriculum to produce competent graduates. Throughout their education, computer engineering students should encounter different approaches to design so that they become familiar with the strengths and weaknesses of these approaches. Typically, the context in which design occurs provides a framework to decide which choices one must make. Depending on the specific application requirements, the design context may emphasize technical considerations, reliability, security, cost, user interface, or other considerations. Development of the requisite design skills cannot be achieved through a single course, but must be integrated throughout the curriculum, building on both the students' accumulated technical knowledge and prior design experiences.

One area of particular concern to the computer engineer is the software/hardware interface where difficult trade-off decisions often provide engineering challenges. Considerations on this boundary lead to an appreciation of and insights into computer architecture and the importance of a computer's machine code. At this boundary, difficult decisions regarding hardware/software trade-offs can occur and this leads naturally to the design of special purpose computers and systems. For example, in the design of a critical-safety system, it is important to ensure that the system not harm the user or the public. The computer engineer must thoroughly test, even with unlikely parameters, the hardware and software, and ultimately the system itself, to ensure the proper and reliable operation of the system.

At a different level, there are all the difficult issues of software design, including the human-computer interface. Addressing this comprehensively can lead to considerations about multi-media, graphics, animation, and a whole host of technologies. Similarly, one can make the same argument for issues in hardware design. In short, design is central to computer engineering.

5.2.2 The Culminating Design Experience

The concept of a culminating design project is widely valued as an important experience that occurs toward the end of a curriculum. Students consider a significant problem associated with a discipline and, in solving the problem, they have the opportunity to demonstrate their ability to provide a solution. Typically, the solution must involve the design and implementation of some product containing hardware and/or software components. The design experience often includes cross-disciplinary teams, which best reflects industry practice. Ideally, the design experience should incorporate engineering standards and realistic constraints to represent what may occur in a real environment.

The culminating design experience should provide students with a wealth of learning benefits. The benefits stemming from this experience include:

- Demonstration of the ability to integrate concepts from several different subjects into a solution
- Demonstration of the application of disciplines associated with computer engineering
- Production of a well-written document detailing the design and the design experience
- Demonstration of creativity and innovation
- Development of time management and planning skills
- Self-awareness opportunities provided by an assessment of achievement as part of a final report

Depending on the approach to assessment, other opportunities arise. Assessment may include a demonstration, a presentation, an oral examination, production of a web page, industry review, and many other interesting possibilities. Although not listed in the core body of knowledge, the culminating design experience must be an integral part of the undergraduate experience.

5.3 The Laboratory Experience

The laboratory experience is an essential part of the computer engineering curriculum and serves multiple functions. As in any engineering curriculum, it is important that computer engineering students have many opportunities to observe, explore and manipulate characteristics and behaviors of actual devices, systems, and processes. This includes designing, implementing, testing, and documenting hardware and software, designing experiments to acquire data, analyzing and interpreting that data, and in some cases, using that data to correct or improve the design. A laboratory setting most effectively demonstrates such experiences either as an integral part of a course or as a separate stand-alone course.

Introductory laboratories are somewhat directed and designed to reinforce concepts presented in lecture classes and homework. Such activities demonstrate specific phenomena or behavior, and provide experiences with measuring and studying desired characteristics. Intermediate and advanced laboratories should include problems that are more open-ended, requiring students to design and implement solutions, to design experiments to acquire data needed to complete the design or measure various characteristics.

Laboratories should include some physical implementation of designs such as electronic and digital circuits, bread-boarding, microprocessor interfacing, prototyping, and implementation of hardware and software.

Laboratories should also include application and simulation software to design small digital and computer systems. The use of simulation tools to model and study real systems is often desirable and necessary to allow students to study systems that are not practical to design and implement physically. Such tools would also be useful where it might be difficult to acquire the detailed information necessary to study their behavior.

Students should learn to record laboratory activity to document and keep track of all design activities, conducted experiments, and their measured/observed results whether good or bad. It also offers opportunities to record trade-offs and to explore the effects of those design tradeoffs. The laboratory experience should also assist students in learning practical issues, such as the following:.

- Safety in all laboratories, especially where electronic equipment and electricity pose dangers
- Proper use of computers and other test equipment
- Building electronic circuits and devices
- Understanding the processes and concerns associated with product development and manufacturing
- Recognizing opportunities for trade-offs and being able to resolve decisions in this area; the trade-off between hardware and software is of particular concern
- Treating laboratories as places of serious study and endeavor

At the formative stages of their education, students often are motivated by the "hands-on" nature of engineering. The laboratory experience capitalizes on this interest to provide a foundation for other important elements of practical activity. Fundamentally, carefully planned practical assignments in a laboratory setting should help students develop confidence in their technical ability. The laboratory experience should help students develop the expertise to build new devices and to appreciate the important role of technical staff, workshop teams, and professionals from other disciplines.

5.4 The Role of Engineering Tools

The use of tools is fundamental to engineering to effectively organize information and manage design complexity. Familiarity with commonly used tools, the ability to deploy them in appropriate situations, and the ability to use them effectively are important skills. Recognizing the potential for tool use is a highly valued skill and in non-standard contexts can provide important insights. In the rapidly changing world of computer engineering, there are opportunities for identifying roles for new tools. The development and exploitation of high quality tools is part of the role of the computer engineer.

For the computer engineer, the relevant range of tools spans the whole hardware and software spectrum. Hardware design and analysis tools include instruments for measuring and analyzing hardware behavior, VLSI design software, hardware description language and other design modeling tools, simulators and emulators, and debugging tools. Other hardware tools include those to support circuit design, printed circuit design layout, analyzing circuit behavior, block diagrams creation and editing, modeling communications systems, modeling mixed analog and digital simulation, design rule checking, and virtual instruments. Software design and analysis tools include operating systems, editors, compilers, language processors, debuggers, and computer-aided software engineering (CASE) tools. General support tools include mathematical analysis programs (e.g. MATLAB, MathCad), office software (word processors, spreadsheets, browsers, and search engines), databases, communications software, and project management tools.

Not every computer engineering program will incorporate all of these tools. The program should incorporate appropriate tools throughout the program of study, consistent with the program's goals and objectives. Identifying the scope for the development of tools and components generally is yet another role for the computer engineer. A natural subsequent activity is engaging in the design and development of these. Such activities need to be guided by concerns for quality in all its different guises – safety, usability, reliability, and so on.

5.5 Applications of Computer Engineering Principles

Given the nature of computer engineering and the expectations of students entering such courses, applications play a fundamental role. Instructors can use applications as a means for:

- Motivating students in their studies
- Guiding their thinking and ambition
- Providing justification for the inclusion and the prominence of certain material
- Demonstrating the application of theoretical ideas

A program can achieve these attributes through a whole range of possible routes. These include the use of up-to-date and topical case studies, guided reading, assessments, speakers from industry, and other diverse paths. This experience can happen at a whole range of levels including chip design, software tools, and entire systems. Suitable applications can also provide a forum for group work, perhaps of an interdisciplinary nature. To this end, all computer engineering students should engage in an in-depth study of some significant application that uses computing engineering in a substantive way.

Computer engineering students will typically have a wide range of interests and professional goals. For many students, in-depth study of some aspect of computer engineering will be extremely useful. Students might accomplish such work in several ways. Some approaches might include an extended internship experience or the equivalent of a full semester's work that would count toward a major in that discipline. Some institutions offer cooperative education programs in which students alternate terms of study and engineering work in industry. Activities of this kind can be interdisciplinary in nature and provide opportunities for particularly beneficial kinds of group activity. Thus, the computer engineer may have to work with professionals from other disciplines, which may include computer scientists, electrical engineers, financial experts, marketers, and product designers.

5.6 Complementary Skills

With the relatively recent worldwide expansion in higher education, there are pressures on institutions to ensure that graduates have the capacity to meet the needs of employers. Indeed, in many ways a more positive view is that institutions appear as agents of change capable of moving into employment with skills and expectations that ensure that organizations benefit from their presence and involvement.

One aspect of this is to ensure that students possess a set of transferable or personal skills such as communication skills, group working skills, and presentational skills. Transferable skills are those skills a person can use in any occupation and can convey them from one type of work to another without retraining. Additionally, one could include library and research skills as well as professional skills such as time management, project management, information management, career development, self-awareness, and keeping up-to-date with innovations in the field. From a motivational perspective, one should assess these skills in the context of computer engineering and in a manner that highlights their relevance and importance to the discipline.

There is always a danger that time spent on complementary skills can absorb excessive amounts of time and effort and swamp or displace the more traditional material, thereby reducing knowledge. There are delicate issues of balance here, and typically, a subtle approach to both teaching and assessment is required to ensure that there is not imbalance in the curriculum.

5.7 Communication Skills

Computer engineers must be able to communicate effectively with colleagues and clients. Because of the importance of good communication skills in nearly all careers, students must sharpen their oral and writing skills in a variety of contexts—both inside and outside of computer engineering courses.

One particular aspect of the activity of a computer engineer is to pass project requirements to a workshop or to technical support staff, which in an industrial setting may be local or remote. Providing clear and succinct instructions and having a proper regard for the role and purpose of support staff affects the efficiency and the nature of the working environment. This trait is a fundamental communication skill. Considering these issues, students should learn to:

- Communicate ideas effectively in written form; this should include technical writing experiences (e.g. of specifications, requirements, safety cases, documentation) as well as report writing and this should address the use of figures, diagrams and appropriate references
- Make effective oral presentations, both formally and informally
- Understand and offer constructive critiques of the presentations of others
- Argue (politely yet effectively) in defense of a position
- Extract requirements from a customer by careful and penetrating questions using a disciplined and structured approach
- Demonstrate the capabilities of a product

While institutions may adopt different strategies to accomplish these goals, the program of each computer engineering student must include numerous occasions for improving these skills in a way that emphasizes writing, speaking, and active listening skills.

To enhance or emphasize the requisite communication skills needed by all students, a computer engineering curriculum at a minimum should require:

- Course work that emphasizes the mechanics and process of writing
- Course work that emphasizes the mechanics and process of speaking
- One or more formal written reports
- Opportunities to critique a written report
- One or more formal oral presentations to a group
- Opportunities to critique an oral presentation

Furthermore, the computer engineering curriculum should integrate writing and verbal discussion consistently in substantive ways. Institutions should not view communication skills as separate entities; instead, teachers should incorporate fully such skills into the computer engineering curriculum and its requirements.

A complementary and important set of communication skills arise in the context of electronic media. Increasingly these have a central role to play in the life of the engineer. Apart from the obvious need to address areas such as email and web design, students should engage at some level the ideas on effective cooperative working and group learning, which have an increased prominence in the curriculum.

5.8 Teamwork Skills

Few computer engineering professionals can expect to work in isolation for very much of the time. Major computer engineering projects are often, if not always, implemented by groups of people working together as a team. Many times the teams are interdisciplinary in nature. Computer engineering students therefore need to learn about the mechanics and dynamics of effective team participation as part of their undergraduate education. Moreover, because the value of working in teams (as well as the difficulties that arise) does not become evident in small-scale projects, students need to engage in team-oriented projects that extend over a reasonably long period of time, possibly a full semester or a significant fraction thereof.

Many of the problems of teamwork relate to communication skills. Where multi-disciplinary teams are involved, individuals tend to receive roles, at least in part, based on their technical expertise. In team activity, however, there are important additional issues related to such matters as the nature and composition of teams, roles within teams, organizing team meetings, developing methods of reaching consensus and for recording decisions, the importance of interfaces, the nature of deadlines and planning, and the importance of quality control mechanisms. Computer engineering programs should include activities that ensure students have the opportunity to acquire these skills as undergraduates; for example:

- Opportunities to work in teams beginning relatively early in the curriculum
- A significant project that a small student team undertakes that involves a complex design and implementation of some product or prototype

5.9 Lifelong Learning Skills

Rapid technological change has been a characteristic of computer engineering and is likely to remain so for some time to come. Graduates must be able to keep up-to-date with that change and a key requirement of undergraduate education is to equip them with the mechanisms for achieving this.

A number of basic strategies seem appropriate. First, the curriculum itself must be up-to-date, the equipment has to be up-to-date, and faculty members need to be engaged in relevant scholarship. Relevant reference material such as textbooks, software, web sites, case studies, and illustrations can be part of the learning experience with the aim of identifying sources of up-to-date and interesting information. In addition, more considerations are fundamental.

Lifelong learning is essentially an attitude of mind. Institutions can foster such attitudes by novel approaches to teaching and learning that continually question and challenge situations and by highlighting opportunities for advances. Instructors can challenge students by assessments and exercises that seek to explore new avenues. It is also essential to see learning as an aspect that merits attention throughout the curriculum. It is possible to have a planned learning experience that challenges student thought processes. Table 4.4 suggests stages that [Fellow, 2002] identified in which learning is possible and the manner of participation by student and teacher.

Table 4.4
Learning Stages

Stage	***Student***	***Instructor***	***Instructional Example***
1	Dependent	Authority/coach	Lecture, coaching
2	Interested	Motivator/guide	Inspirational lecture, discussion group
3	Involved	Facilitator	Discussion lead by instructor who participates as equal
4	Self-directed	Consultant	Internships, dissertation, self directed study group

5.10 The Business Perspective

To complement the technical side of their experiences, computer engineer needs to have an understanding of the various non-technical processes associated with the development of new products. Fundamentally, the computer engineer needs to develop an appreciation of creativity and innovation and have an eye to new opportunities for the creation of wealth, both within established companies and in entrepreneurial ventures. Students can benefit from such knowledge in multiple ways, including:

- Understanding the importance of the financial and economic imperatives associated with new products and organizations
- Appreciating the relevance of the marketing perspective
- Knowing what is involved in product design and product acceptability
- Appreciating the benefits of teamwork, often multi-disciplinary in nature

In addition, students need to appreciate their fiscal responsibilities to their employers. Time translates to money and the importance to complete jobs on schedule becomes important. The business world can also present trade-offs between corporate needs and ethics. Students should be aware of the professional challenges that may await them in government or corporate service.

Within the computer engineering curriculum, such topics may be covered in separate courses (for example, economics, engineering economics, marketing, or accounting), included as part of the culminating design project, or integrated into other courses throughout the program.

5.11 The Elements of an Engineering Education

In summary, proper preparation for professional practice should result in graduates who are capable of the following:

- Seeing their discipline as based on sound principles and sound underpinnings, to recognize what these are, and to be able to apply them
- Understanding the important relationship between theory and practice
- Placing importance on design and being able to select appropriate approaches in particular contexts
- Recognizing the importance of understanding the relevant professional, ethical, and legal issues and the framework
- Recognizing the importance of tools; being able to respond to the challenges of building them and recognizing the need to use these properly and effectively
- Recognizing the range of applications for their work
- Seeing innovation and creativity as important and understanding relevant business perspectives and opportunities
- Recognizing the importance of team activity and the strengths that can be derived from this
- Understanding principles of product design including health and safety as well as marketing issues
- Seeing disciplined approaches as being important
- Understanding the social context within which engineers needs to operate
- Being able to address a significant problem in computer engineering, and demonstrating the ability to deploy an appropriate selection of tools and techniques as well as a disciplined approach in arriving at a solution of the problem

Beyond these characteristics, this chapter has sought to address the range of basic ingredients that institutions must assemble and carefully integrate into a computer engineering program to ensure that graduates are aware of the best traditions of engineering practice.

Chapter 6

Professionalism

One aspect that makes computer engineers different from other computing specialists is their concentration on computer systems that include both hardware and software. Computer engineers design and implement computing systems that often affect the public and 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, 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 global implications of their activities.

6.1 Introduction

The social context of engineering should be an integral component of engineering design and development. The public would not expect that the design and construction of a building, bridge, or tunnel would be void of social context. Likewise, it would not expect that the design and construction of a computer system used in an x-ray machine would be void of that same context. Computer engineers should apply best practices to their work. They should also follow prescribed rules of professional practice and not engage in activities that would tarnish their image or that of their practicing colleagues.

Professionalism and ethics should be the cornerstone of any curriculum in computer engineering. The focus on design and development makes social context paramount to one's studies in the field. Professionalism should be a constant theme that pervades the entire curriculum. Computer engineering students must learn to integrate theory, professional practice, and social constructs in their engineering careers. Computing professionalism should be a major emphasis of the curriculum.

6.2 Decisions in a Societal Context

As designers of computing systems, computer engineers will face many decisions in their careers. While most of these decisions will be technical ones, others will involve a significant societal context. Computer engineers should understand the legal ramifications of contract law, business organization and management, and corporate law.

Of particular importance are issues related to intellectual property. An understanding of patent law is important, particularly when the companies for whom they work may have an active patent program. It is also necessary to understand copyrights since many employers copyright the software they produce. Another method of protecting intellectual property is the use of trade secrets. Different governments have different laws regarding patents, copyrights, and trade secrets. Since the computer engineer will be working in a global context, an understanding of patents, copyrights, and trade secrets and their application is important.

The topics of privacy and secrecy are fundamental to computing. Computers can store vast amounts of information about individuals, businesses, industries, and governments. People can use this information to create profiles of these entities. Computer engineers who are involved in the design of information storage systems must be cognizant of the multiple uses of the systems they develop. Computer engineering students should study cases that trigger an awareness of the social context of how information systems maybe used.

Computer engineers will most certainly have to deal with tradeoffs. Sometimes these are technical decisions such as time versus space tradeoffs in a computer system. Sometimes, however, they involve social, economic, or ethical tradeoffs. Such decisions can be about levels of risk, product reliability, and professional accountability. Computer engineers must be aware of the ramifications of taking risks, be aware of the social consequences, be accountable for the designs they develop, and be aware of the actions they take. These decisions may even involve safety critical systems or life/death situations. Good engineers should not only be cognizant of the societal effects of such decisions, but they should take measures to act professionally to protect the public and to nurture the public trust.

Best practices begin in the instructional laboratory. Educational institutions should encourage behavioral patterns in laboratories that reflect best practices. Such patterns set a level or norm of behavior and elevate the professional expectations of students. They also create a learning environment that is supportive of the professional tenets to which computer engineers aspire. For example, institutions should establish safety guidelines for the proper use of machines and equipment. Institutions should also provide guidelines on interpersonal skills between students, students working in groups, and students interacting with technicians in a laboratory setting. Institutions should instill a sense of professionalism and best practices in all computer engineering students.

Morality is another aspect of making decisions in a societal context. A computer engineer should be aware that many systems of morality exist. Case studies can be helpful to students so they understand the environments in which they will have to function.

6.3 Fostering Professionalism

The issues highlighted in the previous sections have led many professional societies to develop codes of ethics and professional practice for their constituencies. These codes help practitioners to understand expected standards of professional conduct and the expectation among member practitioners. These codes also provide public information concerning the precepts considered central to the profession. These codes provide a level playing field for professionals with the prospects of avoiding ethical dilemmas whenever possible and helping professionals "do the right thing" when faced with ethical decision making during their course of professional practice. In computing, these codes are often binding upon the members of a society and they provide guidance in helping professionals make decisions affecting their practice. Some of these codes include:

- National Society of Professional Engineers - *NSPE Code of Ethics for Engineers* [NSPE 2003]
- Institute of Electrical and Electronic Engineers (IEEE): *IEEE Code of Ethics* [IEEE 2001]
- Association for Computing Machinery (ACM): *ACM Code of Ethics and Professional Conduct* [ACM 2001]
- ACM/IEEE-Computer Society: *Software Engineering Code of Ethics and Professional Practice* [ACM/IEEECS 1999]
- International Federation for Information Processing (IFIP): *Harmonization of Professional Standards* and also *Ethics of Computing* [IFIP 1998]
- Association of Information Technology Professionals (AITP): *AITP Code of Ethics* and the *AITP Standards of Conduct* [AITP 2002]

Computer engineers can use the codes of these societies to guide them to make decisions in their engineering careers.

Although each of these codes focus on the particular purposes of the society or societies sponsoring them, common themes pervade all of them. Fundamental to all these codes are the responsibilities of the computing and engineering professional to the public and to the public good. Additionally, these codes address issues of conflicts of interest, scope of competence, objectiveness and truthfulness, deception, and professional conduct.

The precepts delineated within these codes should be the hallmark of all practicing computer engineers. Computer engineers should adopt the tenets of these codes of ethics and professional practices in all the work they do. It is incumbent upon educational programs to educate computer engineers to embrace these tenets for the benefit of their own careers and for the benefit of the computing and engineering professions.

The inclusion of professional ethics in a computing engineering curriculum is fundamental to the discipline. A listing of topics appears under the social and professional issues (CE-SPR) area as part of the body of knowledge for computer engineering (see Appendix A).

6.4 Summary

Computer engineers have shaped much of the technology we use today. Indeed, computer engineers will continue to involve themselves to shape the technology we use in the future. The computer engineer must apply the principles of best practices in designing and developing new technologies. Computer engineers should be aware of the dilemmas they might face and they must weigh the options in responding to these dilemmas. Using codes of ethics is a concrete approach to avoid potential problems and to resolve those that exist.

Additionally, computer engineers should understand that the technology they design may affect not only a small group of people, but *all* of society. For example, a company could design a product in one country, develop the product in a second country, and manufacture it in a third company. People from those and many other countries could use it. Computer engineers may be involved in all aspects of the product -- from its design to its delivery. Therefore, computer engineers should be sensitive to the customs and laws affecting those people involved in the entire process.

Computer engineers must be aware of entrepreneurial and business developments and the importance of accounting, marketing, and finance. Many computer engineers will become project leaders; in that setting, they must develop an understanding in the management of multi-disciplinary teams and working groups in industry and government. Levels of such responsibility are part of being a professional and should be continuously cultivated throughout one's studies and career.

It is incumbent upon all computer engineers to uphold the tenets of their profession and to foster the codes of professional practice for their colleagues or students.

Chapter 7

Curriculum Implementation Issues

The creation of a complete degree program (an entire program of study) is far from straightforward. The body of knowledge provides a starting point, but many other influences contribute to the creation of the curriculum. The purpose of this chapter is to explore issues in the design and creation of a complete computer engineering degree program. These issues include specifics such as packaging material from the BOK into courses, determining required mathematics and science courses, and more general considerations such as creating an overall style or ethos for a particular computer engineering degree program.

7.1 General Considerations

A computer engineering program requires a great variety of knowledge, practical skills, transferable skills, and attitudes that need consideration within the one single framework. A program should exhibit an obvious and consistent ethos that permeates a complete program of study. Students who enjoy and respond to particular approaches can be confident that they will continue to enjoy and be successful at the more advanced levels.

One key issue is how to distribute, among the fours years of study, relatively settled material (e.g., circuits or supporting mathematics courses) versus material that is more recent. Computer engineering is a discipline in which the rate of change is very swift and this is likely to remain so. Traditional approaches to course design suggest that fundamental and core material should appear at the start of a program. By its very nature, the logic of this is that this material should exhibit a level of permanence and durability and should be unlikely to change over the lifetime of the program. Then students can build on these foundations as they move forward to the later parts of the program and continue as lifelong learners.

This view requires tempering by consideration of the students' point of view. Students who choose to study computer engineering are often motivated by the hands-on nature of engineering, as well as their prior experience with computers. During their initial academic terms, if students only take courses on mathematics and science, without obvious computer engineering applications, it may create a situation of frustration and disillusionment.

It is desirable to position topics involving very new topics in the later years. These new topics are often at the forefront of research and development and after studying them, students can genuinely claim to be up-to-date in their subject area. That is important since they enter industry or employment as the agents of technology change and transfer. Other considerations will also influence the characteristics of a particular degree program. These considerations include:

- Local needs (institutional or regional)
- Needs of an increasingly diverse student population, and
- Interests and background of the faculty

In some cases, an institution may want to design a computer engineering degree program that focuses on one specific area of computer engineering or perhaps gives students a choice among a few such areas. A variety of specialized degree programs is perfectly achievable within the general framework. Included, for example, would be degrees with particular orientations in areas such as computer communications, embedded computer systems, system level integration, mobile computing systems, computer systems design, computer devices, digital signal processing, multi-media systems, computing and broadcasting, pervasive computing, high integrity computing systems, and real-time systems.

Another consideration is how many modules can be designed specifically for computer engineering students and how many will be shared with either (or both) of computer science or electrical engineering curricula. For instance, institutions may construct a computer engineering curriculum with one of the following alternative options.

- There may be enough students in computer engineering to justify the provision of specialist courses devised solely for computer engineering students
- Alternatively, computer engineers might attend classes offered from the computer science and electrical engineering curricula with additional selected classes being mounted specifically to address the specialist topics for computer engineering students
- Additional possibilities also exist depending on local arrangements and circumstances

7.2 Basic Computer Engineering Components

In assembling the curriculum, institutions must package material into modules, typically into classes or courses. Different institutions will possess different conventions about classes. In keeping with the spirit of the Computer Science Report, the Task Force suggests that program designers think in terms of introductory, intermediate, and advanced classes in computer engineering. These need to encompass and reflect the elements of the engineering ethos identified in Chapter 5 as well as the requirements of the professional, legal and ethical issues outlined in Chapter 6.

7.2.1 Introductory Courses and the Core

It is important to ensure that the curriculum includes at least the minimum coverage specified in the core of the body of knowledge. The core itself does not constitute a curriculum. The Computer Engineering Task Force wished to allow different institutions to devise different and novel curricula that would incorporate the core in different and varied ways.

Introductory courses are the first courses that students encounter and are extremely important. Almost of necessity, they will tend to focus on material from the core and will tend to be compulsory. However, institutions wishing to address the specific needs of students who already have considerable experience and competence in core material (e.g. of programming) may permit some form of recognition of this experience.

7.2.2 Intermediate Courses

By their very nature, intermediate courses provide a bridge between introductory courses and advanced courses. They may well include core material but could also include material that falls outside the core. Intermediate courses will typically have introductory courses or other intermediate courses as prerequisites. Typically, these courses occur at second and third year level. Students may have a choice of intermediate courses, but such choices are likely to be limited.

7.2.3 Advanced Courses

The term “advanced course” should mean those courses whose content is substantially beyond the material of the core. The knowledge units give testimony to the rich set of possibilities that exist for these. Institutions will wish to orient such courses to their own areas of expertise, guided by the needs of students, the expertise of faculty members and the needs of the wider community. They will reflect leading edge developments and reflect the stated orientation of the degree program. However, if specific core units are not included in the introductory and intermediate phase, the institution must then ensure that students acquire this material in advanced courses. Institutions should give students a reasonable choice of advanced courses so that they can specialize in areas of choice, consistent with program objectives.

7.2.4 Culminating Project

The culmination of the study of computer engineering should include a final year project that requires students to demonstrate the use of a range of knowledge, practices and techniques in solving a substantial problem. This culminating experience can synthesize a broad range of undergraduate learning and can foster teamwork and professional practice among peers. The culminating project is essential to every computer engineering program.

7.2.5 Engineering Professional, Ethical, and Legal Issues

The curriculum must address the elements of the engineering ethos as well as professional, legal, and ethical issues with progression and integration taking place within these elements as well as within the technical domain. Addressing this vast array of requirements presents a complex task. If an institution treats the various requirements separately and in an undisciplined fashion, the result will be less than satisfactory.

Earlier, mention was made of the importance of giving attention to creativity and innovation in a computer engineering context. It is worth remarking that certain approaches to the other important matter of professional, legal, and ethical matters can have the highly undesirable effect of stifling beneficial innovation. Teachers need to recognize this and indeed take positive steps to counter such trends. It is most important to ensure that the balance is heavily in favor of beneficial innovation and creativity.

A program may choose to include courses on topics such as ethics, business, or legal issues taught by specialists in those fields. However, such courses do not eliminate the need to address these topics in the context of computer engineering.

7.2.6 Communication Skills

Students in computer engineering must be able to communicate ideas effectively in writing and in both formal and informal oral presentations. Therefore, computer engineering programs must develop in their students the ability to present both technical and non-technical material to a range of audiences using rational and reasoned arguments. The manner of presentation includes oral, electronic, and written methods that are necessary for all engineering programs. While courses taught outside of computer engineering may contribute to achieving these skills, it is essential that appropriate communication requirements be included in computer engineering courses. This is necessary to ensure that students have the ability to communicate discipline-specific content; further, such activities contribute to the students' learning of technical material.

7.2.7 Assessment of Student Learning

One should observe a number of important considerations in the assessing of students' learning beyond those that apply to all university learning.

- There is the issue of coursework; many topics lend themselves naturally to practical laboratory work. It is normally desirable to ensure that the practical work counts towards the final assessment; indeed some would take the view that a pass in the practical activity should be mandatory for a pass overall. All aspects of the practical activity must be of high quality
- Where there are sophisticated technical skills involved, there should be sufficient time provided for laboratory experiences with support for the students to ensure that they are learning the material and acquiring effective skills.

When assessing transferable skills, there is merit in integrating this assessment with the assessment of computer engineering activity. In this manner, the skills manifest themselves in their natural setting and students learn ways to address them. An additional advantage of this approach is that it serves to reduce the assessment load.

7.3 Courses Often Taught Outside of Computer Engineering

Beyond the technical courses specifically on computer engineering, a number of other courses reflect material that needs inclusion within the curriculum. For example, computer engineering students must learn a certain amount of mathematics and science, which form the basis for engineering. In this subsection, we discuss various materials that students must learn, but that typically appear in courses outside of the department where computer engineering resides.

7.3.1 Mathematics Requirements

Mathematical techniques and formal mathematical reasoning are integral to most areas of computer engineering. The discipline depends on mathematics for many of its fundamental underpinnings. In addition, mathematics provides a language for working with ideas relevant to computer engineering, specific tools for analysis and verification, and a theoretical framework for understanding important ideas.

Given the pervasive role of mathematics within computer engineering, the curriculum must include mathematical concepts early and often. Basic mathematical concepts should appear early within a student's course work and later courses should use these concepts regularly. While different colleges and universities will need to adjust their prerequisite structures to reflect local needs and opportunities, it is important for upper-level computer engineering courses to make use of the mathematical content developed in earlier courses. A formal prerequisite structure should reflect this dependency.

Some material that is mathematical in nature lies in a boundary region between computer science and engineering and computer engineering faculty members may actually teach it. Other material such as basic differential and integral calculus will likely be under the purview of faculty members outside the department where computer engineering resides. For example, discrete structures topics are important for all students in computer engineering and the Task Force considers it as much as part of computer engineering as mathematics or computer science. Regardless of the implementation, computer engineering programs must take responsibility for ensuring that students obtain the appropriate mathematics they need.

The Computer Engineering Task Force makes the following recommendations with respect to the mathematical content of the computer engineering curriculum.

- *Discrete structures*: All students need exposure to the tools of discrete structures. All programs should include enough exposure to this area to cover the core topics specified in the computer engineering body of knowledge.
- *Differential and integral calculus*: The calculus is required to support such computer engineering material as communications theory, signals and systems, and analog electronics and it is fundamental to all engineering programs.
- *Probability and statistics*: These related topics underpin considerations of reliability, safety, dependence, and various other concepts of concern to the computer engineer. Many programs will have students take an existing course in probability and statistics; some programs may allow some students to study less than a full semester course in the subject. Regardless of the implementation, all students should get at least some brief exposure to discrete and continuous probability, stochastic processes, sampling distributions, estimation, hypothesis testing, and correlation and regression.
- *Additional mathematics*: Students should take additional mathematics to develop their sophistication in this area and to support classes in topics such as communications theory, security, signals and systems, analog electronics. That mathematics might consist of courses in any number of areas, including further calculus, differential equations, transform theory, linear algebra, numerical methods, complex variables, geometry, number theory, or symbolic logic. The choice should depend on program objectives, institutional requirements, and the needs of the individual student.

7.3.2 Science Requirements

The process of abstraction (data collection, hypothesis formation and testing, experimentation, analysis) represents a vital component of logical thought within the field of computer engineering. The scientific method represents a basis methodology for much of the discipline of computer engineering, and students should have a solid exposure to this methodology.

Computer engineering students need a knowledge of basic sciences, such as physics and chemistry. Basic physics concepts in electricity and magnetism for the basis for much of the underlying electrical engineering content in the body of knowledge. Other science course, such as biology, are relevant to specific application areas in which computer engineers may specialize. The precise nature of the basic science requirement will vary, based on institutional and programs needs and resources.

To develop a firm understanding of the scientific method, students must have direct hand-on experience with hypothesis formulation, experimental design, hypothesis testing, and data analysis. While a curriculum may provide this experience as part of the basic science coursework, another way of addressing this is through appropriate courses in computer engineering itself. For example, considerations of the user interface provide a rich vein of experimental situations.

It is vital that computer engineering students "do science" and not just "read about science" in their education. The overall objectives of this element of the curriculum include the following:

- Students should acquire knowledge of the basic sciences underlying computer engineering and relevant application areas.
- Students must develop an understanding of the scientific method and experience this mode of inquiry in courses that provide some exposure to laboratory work, including data collection and analysis.
- Students may acquire their scientific perspective in any of a variety of domains, depending on program objectives and their area of interest.

7.3.3 General Education

Most institutions have a general education requirement that applies to all students in all disciplines. The size and content of this requirement varies widely, depending on institutional mission, legal requirements, and other factors. General education courses often include subjects drawn from the humanities, social sciences, languages, and the liberal arts. In designing a computer engineering program, attention should be given to utilizing these course requirements to contribute to the students' understanding of the social context of engineering and the potential impact of engineering solutions in a global environment.

7.4 Degree Program Implementation: Strategies and Examples

Institutions that wish to follow the suggestions provided herein will typically begin by choosing an implementation for the introductory phase and an implementation for the intermediate phase. From there, they will choose advanced elective courses that conform to local conditions and program objectives. The following attempts to assist institutions to fulfill their program objectives for computer engineering.

7.4.1 Course Considerations

As previously mentioned, the precise courses will depend on the character of each individual program of study. However, in broad terms various considerations will tend to govern the courses at the introductory, intermediate, and advanced levels.

At the initial stages, it is appropriate to develop basic skills within introductory courses. Accordingly, introductory courses should address the following characteristics.

- Basic skills in the design and development of a range of electronic circuits and digital systems
- Basic skills in programming and algorithmic design
- An understanding of the basic structure and organization of a variety of computer systems

These characteristics should address the basic electronics and chip aspects as well as the software approach. These should serve to integrate the various aspects of the courses and provide an overview of the discipline of computer engineering. Fundamentally, the perspective of the computer system as a hierarchy of abstract machines is relevant to the various approaches one could take and suggests references to alternative models.

At the intermediate level, the program should apply the basic skills already acquired and seek to develop them further. Instructors should indicate how to utilize these skills in the design and the development of various components such as in hardware, software, communications, or hybrid systems. Additional coursework serves to introduce remaining core topics and focus students towards areas of specialization. Again, the choices here will depend heavily on the precise characteristics of the program of study. In developing intermediate courses, it is

important to be aware that skills require constant reinforcing. Thus, as an example, it is typically not desirable to introduce students to programming and then drop programming for several semesters.

7.4.2 Elective Courses

At the advanced level, the Computer Engineering Task Force has identified a range of possible elective courses that focus on material that, in keeping with the spirit of computer engineering, involves both hardware and software at an advanced level. Of course, one recognizes that other courses may concentrate on specific aspects of hardware or software. Table 7.1 identifies some elective courses that likely would be relevant to computer engineering programs. The number and scope of electives will vary widely among programs, based on constituent needs, program goals, and resources.

Table 7.1
Examples of Elective Courses

Fault tolerant computer systems	Performance evaluation	Advanced computer architecture
Digital video processing	System level integration	Audio signal processing
Parallel processing	High performance computer systems	Mobile computer systems
Re-configurable computing	Hardware software co-design	Multi-media signal processing
Intelligent systems	Computer security	Security in wireless systems
Safety critical systems	Tool development	Computer based devices
Pervasive computing	Multimedia systems and algorithms	Novel computer architectures
Advanced graphical systems	Genetic algorithms	Distributed information systems
Computer based medical systems	Entertainment systems	Virtual devices
Virtual environment	Robotics	Multi-valued logic systems
Quantum computing	DNA computing	Nano-computing

7.5 Degree Titles and Organizational Structures

As noted in Section 2 of this report, computer engineering programs are offered under a variety of degree titles and within many different organizational structures. As a general rule, variations in the program title tend to imply variations in program content, while variations in organizational structures tend to affect the manner in which courses are organized and taught. Computer engineering is not centric to any one locale or country. Many institutions have considerable expertise in the design and development of hardware and computer systems and their program provisions reflect this, whether or not the program has the specific title of 'computer engineering'.

Programs of study with a body of knowledge comparable to that defined in this report likely will have titles such as computer engineering or computer systems engineering, Other program titles, such as computer and electronic systems, electrical and computer engineering, or computer science and engineering typically reflect a more broadly based set of concerns (and a corresponding broader body of knowledge) than might be implied by the "computer engineering" title. Such program titles also may reflect joint programs administered by multiple academic departments.

Most computer engineering programs are offered by institutions that also offer other engineering and/or computer science programs. Such institutions, thus, have existing resources that may be applied to support a computer engineering program, whether or not it is administratively managed by those units. All organizational arrangements have both drawbacks and benefits. For example, students may take a blend of courses designed primarily for mainstream computer science or electrical engineering majors with relatively few courses specially designed for computer engineering students. Such a structure will likely affect the topics added to the core elements of the body of knowledge, based on maximizing course commonality rather than other factors. However, such programs may achieve "accredited" status (sometimes by more than one professional body) and produce graduates who are highly attractive to industry specifically because of their breadth of knowledge.

Independent of organizational structure, it is essential that a computer engineering program have a core faculty of appropriate size and technical competence. Many of the technical courses included within a computer

engineering program may be taught by faculty from areas such as computer science, electrical engineering, or physics. However, the distinct disciplinary emphases and the preparation for professional practice requires faculty with appropriate technical training and professional expertise.

7.6 Sample Curricula

Appendix B provides four sample implementations of complete computer engineering programs. To provide a framework for the curriculum that illustrates the ideas presented in this report, the first three examples assume the following.

- Each year consists of two semesters with a student studying five modules (courses) per semester. Each module is approximately 45 contact hours (42 contact hours for presentation and 3 contact hours for assessment).
- Students should experience 3 computer engineering modules in the first year of study, 4 or 5 in the second year of study, and 5 or 6 in each of the third and fourth years of study.

The above pattern is used by many US institutions, and is common in many other parts of the world. The fourth example implementation is of a three-year program (such as commonly exists in the U.K., Europe, and some other countries) and assumes additional pre-university preparation in mathematics, science, and general studies.

Chapter 8

Institutional Challenges

This report provides a significant resource for colleges and universities seeking to develop or improve undergraduate programs in computer engineering. The appendices to this report offer an extensive analysis of the structure and scope of computer engineering knowledge along with viable approaches to the undergraduate curriculum. Implementing a curriculum successfully, however, requires each institution to consider broad strategic and tactical issues that transcend such details. The purpose of this chapter is to enumerate some of these issues and illustrate how addressing those issues affects curriculum design.

8.1 The Need for Local Adaptation

The task of designing a computer engineering curriculum is a difficult one in part because so much depends on the characteristics of the individual institution. Even if every institution could agree on a common set of knowledge and skills for undergraduate education, many additional factors would influence curriculum design. These factors include the following:

- *The type of institution and the expectations for its degree programs*: Institutions vary enormously in the structure and scope of undergraduate degree requirements. A curriculum that works well at a small college in the United States may be completely inappropriate for a research university elsewhere in the world.
- *The range of postgraduate options that students pursue*: Institutions whose primary purpose is to prepare a skilled workforce for the computer engineering profession presumably have different curricular goals than those seeking to prepare students for research and graduate study. Individual schools must ensure that the curriculum they offer gives students the necessary preparation for their eventual academic and career paths.
- *The preparation and background of entering students*: Students at different institutions—and often within a single institution—vary substantially in their level of preparation. As a result, computer engineering departments often need to tailor their introductory offerings so that they meet the needs of their students.
- *The faculty resources available to an institution*: The number of faculty in a computer engineering department may vary from as little as three or four at a small college to 100 or more at a large research university. The flexibility and options available in these smaller programs is obviously a great deal less. Therefore, faculty members in smaller departments need to set priorities for how they will use their limited resources.
- *The interests and expertise of the faculty*: Individual curricula often vary according to the specific interests and knowledge base of the department, particularly at smaller institutions where expertise is concentrated in particular areas.

Creating a workable curriculum requires finding an appropriate balance among these factors, which will require different choices at every institution. No single curriculum can work for everyone. Every college and university will need to consider the various models proposed in this document and design an implementation that meets the need of their environment.

8.2 Principles for Curriculum Design

Despite the fact that curriculum design requires significant local adaptation, curriculum designers can draw on several key principles to help in the decision-making process. These principles include the following:

- *The curriculum must reflect the integrity and character of computer engineering as an independent discipline.* Computer engineering is a discipline in it own right. A combination of theory, practice, knowledge, and skills characterize the discipline. Any computer engineering curriculum should therefore ensure that both theory and a spirit of professionalism guide the practice.

- *The curriculum must respond to rapid technical change and encourage students to do the same.* Computer engineering is a vibrant and fast-changing discipline. The enormous pace of change means that computer engineering programs must update their curricula on a regular basis. Equally importantly, the curriculum must teach students to respond to change as well. Computer engineering graduates must keep up to date with modern developments and the prospects of doing so should stimulate their engineering curiosity. One of the most important goals of a computer engineering program should be to produce students who are life-long learners.

- *Outcomes a program hopes to achieve must guide curriculum design.* Throughout the process of defining a computer engineering curriculum, it is essential to consider the goals of the program and the specific capabilities students must have at its conclusion. These goals—and the associated techniques for determining whether a program is meeting these goals—provide the foundation for the entire curriculum. Throughout the world, accreditation bodies have focused increasing attention on the definition of goals and assessment strategies. Programs that seek to defend their effectiveness must be able to demonstrate that their curricula in fact accomplish what they intended to do.

- *The curriculum as a whole should maintain a consistent ethos that promotes innovation, creativity, and professionalism.* Students respond best when they understand the expectations of them. It is unfair to students to encourage particular modes of behavior in early courses, only to discourage that same behavior in later courses. Throughout the entire curriculum, students should be encouraged to use their initiative and imagination to go beyond the minimal requirements. At the same time, students must be encouraged from the very beginning to maintain a professional and responsible attitude toward their work and give credence to the ethical and legal issues affecting their professional practice.

- *The curriculum must provide students with a culminating design experience that gives them a chance to apply their skills and knowledge to solve challenging problems.* The culmination of an undergraduate computer engineering degree should include a project that requires students to use a range of practices and techniques in solving a substantial problem as a key component in preparing them for professional practice.

8.3 The Need for Adequate Laboratory Resources

It is essential for institutions to recognize that equipment and software costs to support computer engineering programs are large. Software can represent a substantial fraction of the overall cost of computing, particularly if one includes the development costs of courseware. Providing adequate support staff to maintain the laboratory facilities represents another expense. To be successful, computer engineering programs must receive adequate funding to support the laboratory needs of both faculty and students and to provide an atmosphere conducive to learning.

Because of rapid changes in technology, computer hardware generally becomes obsolete long before it ceases to function. The useful lifetime of computer systems, particularly those used to support advanced laboratories and state-of-the-art software tools, may be as little as two or three years. Planning and budgeting for regular updating and replacement of computer systems is essential.

Computer engineering typically has many scheduled laboratories included in the curriculum. The laboratory component leads to an increased need for staff to assist in both the development of materials and the teaching of laboratory sections. This development will add to the academic support costs of a high-quality computer engineering program.

8.4 Attracting and Retaining Faculty

One of the most daunting problems that computer engineering departments face is the problem of attracting qualified faculty. In computer engineering, there are often more advertised positions than the number of highly qualified candidates. The shortage of faculty applicants, coupled with the fact that computer engineers command high salaries outside academia, makes it difficult to attract and retain faculty. Institutions will need to have an

aggressive plan to recruit and retain faculty. Incentives such as hiring packages and modified teaching responsibilities may prove advantageous for this endeavor.

While the computer engineering program may draw on faculty from related disciplines, as a professional field there must be a core faculty with appropriate professional training and experience. Additionally, faculty members must maintain currency with developments in the field. Institutions must make appropriate accommodations for the professional development of faculty, whether achieved through research, conference participation, consulting, or other activities.

8.5 Summary

No single formula exists for success in designing a computer engineering curriculum. Although the Computer Engineering Task Force believes that the recommendations of this report and the specific strategic suggestions in this chapter will prove useful to a wide variety of institutions, every computer engineering program must adapt those recommendations and strategies to match the characteristics of the particular institution. It is, moreover, important to evaluate and modify curricular programs on a regular basis to keep up with the rapid changes in the field. The curricula of the future will depend on the creativity that follows in the wake of this report to build even better computer engineering programs for undergraduates throughout the world.

Endnote References to this Report

[ABET, Design] Definition of Design, ABET 2004-2005 Criteria for Accrediting Programs in Engineering in the United States, Criterion 4
[ABET, 2004] Evaluation Criteria, 2004-2005 Engineering Criteria, <http://www.abet.org/criteria_eac.html>.
[ACM, 1992] ACM Code of Ethics and Professional Conduct, < http://www.acm.org/constitution/code.html>, 16 October 1992.
[ACM/IEEECS, 1999] CAN and IEEE Computer Society, Software Engineering Code of Ethics and Professional Practice, <http://computer.org/certification/ethics.htm>, 1999.
[Aub] CCCE website at <http://www.eng.auburn.edu/ece/CCCE>
[AITP, 2002] Association of Information Technology Professionals, Code of Ethics, <http://www.aitp.org/organization/about/ethics/ethics.jsp >, 2002.
[ASEE'02] American Society for Engineering Education, ASEE Annual Conference and Exhibition, <http://www.asee.org/conferences/annual2002/default.cfm>, Montreal, Canada, 16-19 June 2002.
[ASEE'03] American Society for Engineering Education, ASEE Annual Conference and Exhibition, <http://www.asee.org/conferences/annual2003/default.cfm>, Nashville, Tennessee, 22-25 June 2003.
[ASEE'04] American Society for Engineering Education, ASEE Annual Conference and Exhibition, <http://www.asee.org/conferences/annual2004/default.cfm >, Salt Lake City, Utah, 20-23 June 2004.
[Bennett 1986] W. Bennett. A position paper on guidelines for electrical and computer engineering education. IEEE Transactions in Education, E-29(3):175-177, August 1986.
[EAB 1986] Educational Activities Board. Design education in computer science and engineering. Technical Report 971, Computer Society of the IEEE, October 1986.
[Fellows 2002] Sharon Fellows, Richard Culver, Peter Ruggieri, William Benson *Instructional Tools for Promoting Self-directed Skills in Freshmen*, FIE 2002, Boston, November, 2002.
[FIE'02] Frontiers in Education Conference, <http://www.wpi.edu/News/Conf/FIE2002/>, Boston, Massachusetts, 6-9 November 2002.
[FIE'03] Frontiers in Education Conference, <http://www.fie-conference.org/03/>, Denver, Colorado, 5-8 November 2003.
[FIE'04] Frontiers in Education Conference, <http://www.fie-conference.org/04/>, Savannah, Georgia, 20-23 October 2004.
[IEEE, 1990] IEEE Code of Ethics, <http://www.ieee.org/>, About IEEE, August 1990.
[IFIP, 1998] Harmonization of Professional Standards (Draft Version), <www.ifip.or.at/minutes/C99/C99_harmonization.htm>, October 1998.
[ITEA] International Technology Educational Association, <http://www.iteawww.org/TAA/Glossary.htm>
[ITiCSE'03] Innovation and Technology in Computer Science Education, <http://www.cs.utexas.edu/users/csed/iticse/>, Thessaloniki, Greece, 30 June – 2 July 2003
[ITiCSE'04] Innovation and Technology in Computer Science Education, <http://www.iticse04.leeds.ac.uk/>, Leeds, England, 28-30 June 2004.
[Langdon, et. al. 1986] Design Education in Computer Science and Engineering, Technical Report, IEEE Computer Society Educational Activities Board, October 1, 1986.
[MCSCE'04] International MultiConference in Computer Science and Computer Engineering, <http://www.world-academy-of-science.org:8080/CSREA/ws/>, Las Vegas, Nevada, 21-24 June 2004.
[NSPE, 2003] National Society of Professional Engineers, NSPE Code of Ethics for Engineers, <http://www.nspe.org/ethics/ehl-code.asp>, 2003.
[SIGCSE'03] SIGCSE Technical Symposium, <http://www.csis.gvsu.edu/sigcse2003/>, Reno, Nevada, 19-23 February 2003.
[SIGCSE'04] SIGCSE Technical Symposium, <http://www.csc.vill.edu/sigcse2004/>, Norfolk, Virginia, 3-7 March 2004.
[UKQAA, 2000] Quality Assurance Agency for Higher Education, "Computing, a report on benchmark levels for Computing," Southgate House, Gloucester, England, April 2000.

All References

[Abelson et al, 1985] Harold Abelson and Gerald Jay Sussman with Julie Sussman. *Structure and Interpretation of Computer Programs*. Cambridge, MA: MIT Press, 1985.

[ABET, 2000] Accreditation Board for Engineering and Technology. Accreditation policy and procedure manual. Baltimore, MD: ABET, Inc., November 2000. http://www.abet.org/images/policies.pdf.

[ABET, 2002] Accreditation Board for Engineering and Technology, Inc., "Criteria for Accrediting Engineering Programs," November 2002.

[ABET, 2004] Evaluation Criteria, 2003-2004 Engineering Criteria, <http://www.abet.org/criteria_eac.html>.

[ABET, Design] Definition of Design, ABET 2003-2004 Criteria for Accrediting Programs in Engineering in the United States, Section IV.C.3.d.(3)(c).

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

[ACM, 1968] ACM Curriculum Committee on Computer Science. Curriculum '68: Recommendations for the undergraduate program in computer science. *Communications of the ACM*, 11(3):151-197, March 1968.

[ACM, 1978] ACM Curriculum Committee on Computer Science. Curriculum '78: Recommendations for the undergraduate program in computer science. *Communications of the ACM*, 22(3):147-166, March 1979.

[ACM, 1992] ACM Code of Ethics and Professional Conduct, < http://www.acm.org/constitution/code.html>, 16 October 1992.

[ACM, 1999] ACM Two-Year College Education Committee. Guidelines for associate-degree and certificate programs to support computing in a networked environment. New York: The Association for Computing Machinery, September 1999.

[ACM, 2001] Association for Computing Machinery. ACM code of ethics and professional conduct. New York: The Association for Computing Machinery, May 2001. http://www.acm.org/constitution/code.html.

[ACM/IEEECS, 1999] CAN and IEEE Computer Society, Software Engineering Code of Ethics and Professional Practice, <http://computer.org/certification/ethics.htm>, 1999.

[AITP, 2002] Association of Information Technology Professionals, Code of Ethics, <http://www.aitp.org/organization/about/ethics/ethics.jsp >, 2002.

[APP, 2000] Advanced Placement Program. Introduction of Java in 2003-2004. The College Board, December 20, 2000. http://www.collegeboard.org/ap/computer-science.

[ASEE'02] American Society for Engineering Education, ASEE Annual Conference and Exhibition, <http://www.asee.org/conferences/annual2002/default.cfm>, Montreal, Canada, 16-19 June 2002.

[ASEE'03] American Society for Engineering Education, ASEE Annual Conference and Exhibition, <http://www.asee.org/conferences/annual2003/default.cfm>, Nashville, Tennessee, 22-25 June 2003.

[ASEE'04] American Society for Engineering Education, ASEE Annual Conference and Exhibition, <http://www.asee.org/conferences/annual2004/default.cfm >, Salt Lake City, Utah, 20-23 June 2004.

[Aub] CCCE website at <http://www.eng.auburn.edu/ece/CCCE>

[BCS, 1989a] British Computer Society and The Institution of Electrical Engineers. Undergraduate curricula for software engineers. London, June 1989.

[BCS, 1989b] British Computer Society and The Institution of Electrical Engineers. Software in safety-related systems. London, October 1989.

[Beidler et al, 1985] John Beidler, Richard Austing, and Lillian Cassel. Computing programs in small colleges. Communications of the ACM, 28(6):605-611, June 1985.

[Bennett, 1986] W. Bennett. A position paper on guidelines for electrical and computer engineering education. IEEE Transactions in Education, E-29(3):175-177, August 1986.

[Bott et al, 1991] Frank Bott, Allison Coleman, Jack Eaton, and Diane Rowland. Professional issues in software engineering. London: Pitman, 1991.

[Carnegie, 1992] Carnegie Commission on Science, Technology, and Government. Enabling the future: Linking science and technology to societal goals. New York: Carnegie Commission, September 1992.

[COSINE, 1967] COSINE Committee. Computer science in electrical engineering. Washington, DC: Commission on Engineering Education, September 1967.

[CSAB, 1986] Computing Sciences Accreditation Board. Defining the computing sciences professions. October 1986. http://www.csab.org/comp_sci_profession.html.

[CSAB, 2000] Computing Sciences Accreditation Board. Criteria for accrediting programs in computer science in the United States. Version 1.0, January 2000. http://www.csab.org/criteria2k_v10.html.

[CSTB, 1994] Computing Science and Telecommunications Board. Realizing the information future. Washington DC: National Academy Press, 1994.

[CSTB, 1999] Computing Science and Telecommunications Board. Being fluent with information technology. Washington DC: National Academy Press, 1999.

[Curtis, 1983] Kent K. Curtis. Computer manpower: Is there a crisis? Washington DC: National Science Foundation, 1983. http://www.acm.org/sigcse/papers/curtis83/.

[Davis et al, 1997] Gordon B. Davis, John T. Gorgone, J. Daniel Couger, David L. Feinstein, and Herbert E. Longnecker, Jr. IS'97 model curriculum and guidelines for undergraduate degree programs in information systems. Association of Information Technology Professionals, 1997. http://webfoot.csom.umn.edu/faculty/gdavis/curcomre.pdf.

[Denning et al, 1989] Peter J. Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen B. Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. *Communications of the ACM*, 32(1):9-23, January 1989.

[Denning, 1998] Peter J. Denning. Computing the profession. Educom Review, November 1998.

[Denning, 1999] Peter J. Denning. Our seed corn is growing in the commons. Information Impacts Magazine, March 1999. http://www.cisp.org/imp/march_99/denning/03_99denning.htm.

[EAB, 1983] Educational Activities Board. The 1983 model program in computer science and engineering. Technical Report 932, Computer Society of the IEEE, December 1983.

[EAB, 1986] Educational Activities Board. Design education in computer science and engineering. Technical Report 971, Computer Society of the IEEE, October 1986.

[EC, 1977] Education Committee of the IEEE Computer Society. A curriculum in computer science and engineering. Publication EHO119-8, Computer Society of the IEEE, January 1977.

[Fellows et al, 2002] Sharon Fellows, Richard Culver, Peter Ruggieri, William Benson *Instructional Tools for Promoting Self-directed Skills in Freshmen*, FIE 2002, Boston, November, 2002.

[Feisel and Peterson, 2002] Lyle D. Feisel, George D. Peterson, *Learning Objectives for Engineering Laboratories*, FIE 2002, Boston, November, 2002

[Fleddermann, 2000] C.B. Fleddermann *Engineering Ethics Cases for Electrical and Computer Engineering Students*, IEEE Transactions on Education, vol 43, no 3, 284 – 287, August 2000.

[Feiel et al, 2002] Lyle D. Feisel, George D. Peterson, *Learning Objectives for Engineering Laboratories*, FIE 2002, Boston, November, 2002

[FIE'02] Frontiers in Education Conference, <http://www.wpi.edu/News/Conf/FIE2002/>, Boston, Massachusetts, 6-9 November 2002.

[FIE'03] Frontiers in Education Conference, <http://www.fie-conference.org/03/>, Denver, Colorado, 5-8 November 2003.

[FIE'04] Frontiers in Education Conference, <http://www.fie-conference.org/04/>, Savannah, Georgia, 20-23 October 2004.

[Gibbs et al, 1986] Norman E. Gibbs and Allen B. Tucker. Model curriculum for a liberal arts degree in computer science. Communications of the ACM, 29(3):202-210, March 1986.

[Giladi, 1999] R. Giladi, *An Undergraduate Degree Program for Communications Systems Engineering*, IEEE Transactions on Education, vol 42, no 4, 295 – 304, November 1999.

[Gorgone et al, 2000] John T. Gorgone, Paul Gray, David L. Feinstein, George M. Kasper, Jerry N. Luftman, Edward A. Stohr, Joseph S. Valacich, and Rolf T. Wigand. MSIS 2000: Model curriculum and guidelines for graduate degree programs in information systems. Association for Computing Machinery and Association for Information Systems, January 2000. http://cis.bentley.edu/ISA/pages/documents/msis2000jan00.pdf.

[Gorgone et al, 2002] John T. Gorgone, Gordon B. Davis, Joseph S Valacich, Heikki Topi, David L. Feinstein, and Herbert E. Longenecker, Jr. *IS 2002: Model Curriculum for Undergraduate Degree Programs in Information Systems*, published by the ACM, 2002.

[IEEE, 1990] IEEE Code of Ethics, <http://www.ieee.org/>, About IEEE, August 1990.

[IEEE, 2001] Institute for Electrical and Electronic Engineers. IEEE code of ethics. Piscataway, NJ: IEEE, May 2001. http://www.ieee.org/about/whatis/code.html.

[IFIP, 1998] Harmonization of Professional Standards (Draft Version), <www.ifip.or.at/minutes/C99/C99_harmonization.htm>, October 1998.

[ITEA] International Technology Educational Association, <http://www.iteawww.org/TAA/Glossary.htm>

[ITiCSE'03] Innovation and Technology in Computer Science Education, <http://www.cs.utexas.edu/users/csed/iticse/>, Thessaloniki, Greece, 30 June – 2 July 2003

[ITiCSE'04] Innovation and Technology in Computer Science Education, <http://www.iticse04.leeds.ac.uk/>, Leeds, England, 28-30 June 2004.

[Kelemen et al, 1999] Charles F. Kelemen (editor), Owen Astrachan, Doug Baldwin, Kim Bruce, Peter Henderson, Dale Skrien, Allen Tucker, and Charles Ban Loan. Computer Science Report to the CUPM Curriculum Foundations Workshop in Physics and Computer Science. Report from a workshop at Bowdoin College, October 28-31, 1999.

[Koffman et al, 1984] Elliot P. Koffman, Philip L. Miller, and Caroline E. Wardle. Recommended curriculum for CS1: 1984 a report of the ACM curriculum task force for CS1. *Communications of the ACM*, 27(10):998-1001, October 1984.

[Koffman et al, 1985] Elliot P. Koffman, David Stemple, and Caroline E. Wardle. Recommended curriculum for CS2, 1984: A report of the ACM curriculum task force for CS2. *Communications of the ACM*, 28(8):815-818, August 1985.

[Langdon, et. al. 1986] Design Education in Computer Science and Engineering, Technical Report, IEEE Computer Society Educational Activities Board, October 1, 1986.

[Lee and Messerschmitt, 1998] Edward A. Lee and David G. Messerschmitt. Engineering and education for the future. IEEE Computer, 77-85, January 1998.

[Lidtke et al, 1999] Doris K. Lidtke, Gordon E. Stokes, Jimmie Haines, and Michael C. Mulder. ISCC '99: An information systems-centric curriculum '99, July 1999. http://www.iscc.unomaha.edu.

[Martin et al, 1996] C. Dianne Martin, Chuck Huff, Donald Gotterbarn, Keith Miller. Implementing a tenth strand in the CS curriculum. *Communications of the ACM*, 39(12):75-84, December 1996.

[MCSCE'04] International MultiConference in Computer Science and Computer Engineering, <http://www.world-academy-of-science.org:8080/CSREA/ws/>, Las Vegas, Nevada, 21-24 June 2004.

[Mulder, 1975] Michael C. Mulder. Model curricula for four-year computer science and engineering programs: Bridging the tar pit. Computer, 8(12):28-33, December 1975.

[Mulder and Dalphin, 1984] Michael C. Mulder and John Dalphin. Computer science program requirements and accreditation—an interim report of the ACM/IEEE Computer Society joint task force. *Communications of the ACM*, 27(4):330-335, April 1984.

[Mulder and van Weert, 1998] Fred Mulder and Tom van Weert. Informatics in higher education: Views on informatics and noninformatics curricula. Proceedings of the IFIP/WG3.2 Working Conference on Informatics (computer science) as a discipline and in other disciplines: What is in common? London: Chapman and Hall, 1998.

[Myers and Walker, 1998] J. Paul Myers, Jr. and Henry M. Walker. The state of academic hiring in computer science: An interim review. *SIGCSE Bulletin*, 30(4):32a-35a, December 1998.

[NACE, 2001] National Association of Colleges and Employers. Job outlook '01 (online version). http://www.jobweb.com

[Neumann, 1995] Peter G. Neumann. Computer related risks. New York: ACM Press, 1995.

[Nordheden and Hoeflich, 1999] K.J. Nordheden and M.H. Hoeflich, Undergraduate Research and Intellectual Property Rights, *IEEE Transactions on Software*, vol 19 , no. 5, September / October, 22 – 24, 2002. *Education*, vol 42, no 4, 233-236, November 1999.

[NSF, 1996] National Science Foundation Advisory Committee. Shaping the future: New expectations for undergraduate education in science, mathematics, engineering, and technology. Washington DC: National Science Foundation, 1996.

[NSPE, 2003] National Society of Professional Engineers, NSPE Code of Ethics for Engineers, <http://www.nspe.org/ethics/ehl-code.asp>, 2003.

[NTIA, 1999] National Telecommunications and Information Administration. Falling through the Net: Defining the digital divide. Washington, DC: Department of Commerce, November 1999.

[Nunamaker et al, 1982] Jay F. Nunamaker, Jr., J. Daniel Couger, Gordon B. Davis. Information systems curriculum recommendations for the 80s: Undergraduate and graduate programs. *Communications of the ACM*, 25(11):781-805, November 1982.

[Oklobdzija, 2002] Vojin G. Oklobdzija (editor) *The Computer Engineering Handbook*, published by CRC Press LLC, Florida, USA, 2002.

[OTA, 1988] Office of Technology Assessment. Educating scientists and engineers: Grade school to grad school. OTA-SET-377. Washington, DC: U.S. Government Printing Office, June 1988.

[Paulk et al, 1995] Mark Paulk, Bill Curtis, Mary Beth Chrissis, and Charles Weber. *The capability maturity model: Guidelines for improving the software process*. Reading, MA: Addison-Wesley, 1995.

[QAA, 2000] Quality Assurance Agency for Higher Education. A report on benchmark levels for computing. Gloucester, England: Southgate House, 2000.

[Ralston and Shaw, 1980] Anthony Ralston and Mary Shaw. Curriculum '78—Is computer science really that unmathematical. *Communications of the ACM*, (23)2:67-70, February 1980.

[Richard et al, 1999] W. D. Richard, D. E. Taylor and D. M. Zar, A Capstone Computer Engineering Design Course, *IEEE Transactions on Education*, vol 42, no 4, 288 – 294, November 1999.

[Roberts et al, 2001] Eric Roberts and Gerald Engel (editors) *Computing Curricula 2001: Computer Science*, Report of The ACM and IEEE-Computer Society Joint Task Force on Computing Curricula, Final Report, December 15th, 2001

[Roberts et al, 1995] Eric Roberts, John Lilly, and Bryan Rollins. Using undergraduates as teaching assistants in introductory programming courses: An update on the Stanford experience. *SIGCSE Bulletin* (27)1:48-52, March 1995.

[Roberts, 1999] Eric Roberts. Conserving the seed corn: Reflections on the academic hiring crisis. *SIGCSE Bulletin* (31)4:4-9, December 1999.

[SAC, 1967] President's Science Advisory Commission. Computers in higher education. Washington DC: The White House, February 1967.

[SEEPP, 1998] IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices (SEEPP). Software engineering code of ethics and professional practice (Version 5.2). http://www.acm.org/serving/se/code.htm.

[Shaw, 1985] Mary Shaw. The Carnegie-Mellon curriculum for undergraduate computer science. New York: Springer-Verlag, 1985.

[Shaw, 1991] Mary Shaw and James E Tomayko. Models for undergraduate courses in software engineering. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, January 1991.

[Shaw, 1992] Mary Shaw. We can teach software better. Computing Research News 4(4):2-12, September 1992.

[SIGCHI, 1992] Special Interest Group on Computer-Human Interaction. ACM SIGCHI Curricula for Human-Computer Interaction. New York: Association for Computing Machinery, 1992.

[SIGCSE'03] SIGCSE Technical Symposium, <http://www.csis.gvsu.edu/sigcse2003/>, Reno, Nevada, 19-23 February 2003.

[SIGCSE'04] SIGCSE Technical Symposium, <http://www.csc.vill.edu/sigcse2004/>, Norfolk, Virginia, 3-7 March 2004.

[SWEBOK, 2001] Software Engineering Coordinating Committee. Guide to the Software Engineering Body of Knowledge (SWEBOK). Stone Man Version 0.95. A Project of the IEEE Computer Society, May 2001. http://www.swebok.org/stoneman/version095.html/.

[Tucker et al, 1991] Allen B. Tucker, Bruce H. Barnes, 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, and A. Joe Turner. Computing Curricula '91. Association for Computing Machinery and the Computer Society of the Institute of Electrical and Electronics Engineers, 1991.

[UKQAA, 2000] Quality Assurance Agency for Higher Education, "Computing, a report on benchmark levels for Computing," Southgate House, Gloucester, England, April 2000.

[Walker and Schneider, 1996] Henry M. Walker and G. Michael Schneider. A revised model curriculum for a liberal arts degree in computer science. *Communications of the ACM*, 39(12):85-95, December 1996.

[Zadeh, 1968] Lofti A. Zadeh. Computer science as a discipline. *Journal of Engineering Education*, 58(8):913-916, April 1968.

Appendices

(Each presented in a separate document.)

Appendix A Knowledge Areas with Knowledge Units

Appendix B Sample Curricula

Reviewers

The Computer Engineering Task Force thanks the following individuals for their comments and suggestions in the development of this report.

NAME	AFFILIATION
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy
xx	yyy