Computer Engineering

2004

Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering

A Report in the Computing Curricula Series

Joint Task Force on Computer Engineering Curricula

IEEE Computer Society
Association for Computing Machinery

2004 December 12

This material is based in part on work supported by the
National Science Foundation Grant Number 0229748

Executive Summary

This report presents curriculum guidelines for undergraduate degree programs in computer engineering. It draws upon recent efforts in computing curricula developed by the IEEE Computer Society, the Association for Computing Machinery, and the Association for Information Systems. These efforts resulted in published curricula recommendations in computer science [ACM/IEEECS, 2001], information systems [ACM/AIS, 2002], and software engineering [ACM/IEEECS, 2004]. Recommendations for information technology should appear in 2006.

Computer engineering as an academic field encompasses the broad areas of computer science and electrical engineering. Computer engineering is defined in this report as follows.

> *Computer engineering is a discipline that 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).*

Hence, this unique combination prepares students for careers that deal with computer systems from design through implementation. Computing systems are components of a wide range of products such as fuel injection systems in vehicles, medical devices such as x-ray machines, communication devices such as cell phones, and household devices such as alarm systems and washing machines. Designing computing systems and computing components of products, developing and testing their prototypes, and implementing them to market are examples of what computer engineers would do.

This report provides some background on the computer engineering field and explains how the field evolved. It describes the expectations of graduates of the discipline and shows how those graduates differ from other computing disciplines. It describes the expected background, knowledge, and skills employers expect to see from graduates of computer engineering programs. These include the ability to design computer systems, the realization of the importance of practicing as professionals, and having the breadth and depth of knowledge expected of a practicing engineer. It also discusses how programs in computer engineering may have to stand up to the scrutiny of validation and accreditation by government or private agencies.

The report includes four sample curricula that illustrate a methodology an institution mught use to develop a curriculum in computer engineering based on its locale, mission, and particular goals for its students. The sample curricula are grounded on a fundamental body of knowledge from which an institution may develop a curriculum to fit its needs. This body of knowledge contains broad knowledge areas that are applicable to all computer engineering programs worldwide. Each knowledge area comprises a set of knowledge units. Within each knowledge unit, a set of topics provide the details of study; a set of learning outcomes complements each knowledge unit. Within each knowledge area, some knowledge units are identified as “core”, and should appear in every implemented curriculum, while the remaining knowledge units are

elective. Core units represent the minimal knowledge or depth a program should cover in each area; however, a curriculum containing only core units would not constitute a complete curriculum in computer engineering.

A computer engineering program should contain sufficient coursework at the introductory, intermediate, and advanced levels based on the aforementioned body of knowledge of computer engineering. Programs should augment this coursework by a judicious selection of elective courses that build upon that foundation. Breadth and depth in science and mathematics are important to this discipline. As already mentioned, a design component is essential to the program, and typically culminates with a capstone or senior project. The curriculum should also emphasize professional practice, legal and ethical issues, and the social context in which graduates implement engineering designs. Problem solving and critical thinking skills, oral and written communication skills, teamwork, and a variety of laboratory experiences are essential to the study of computer engineering.

These recommendations support the design of computer engineering curricula that will prepare graduates to function at entry-level positions in industry for continued career growth or to enter graduate programs for advanced study. The recommendations reflect input from industrial and educational institutions. This report is the result of a cooperative effort of the professionals involved. Its intent is to provide interested parties and educational institutions worldwide a flexible way to implement a strong program in computer engineering. We trust that we have achieved that goal.

CE2004 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
Richard C. Jaeger	Auburn 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
N. Ranganathan	University of South Florida
Robert Sloan	University of Illinois – Chicago
Pradip K. Srimani*	Clemson University
Mitchell D. Theys*	University of Illinois – Chicago
Wayne Wolf	Princeton University
Murali Varanasi	University of South Florida

* Primary Authors and Editors of the Final Report

Reviewers

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

NAME	AFFILIATION
Nizar Al-Holou	University of Detroit Mercy, United States
Yury A. Bogoyavlenskiy	University of Petrozavodsk, Russian Federation
Kevin W. Bowyer	University of Notre Dame, United States
Annette Bunker	Utah State University, United States
James M. Conrad	UNC Charlotte, United States
Susan E. Conry	Clarkson University, United States
James A. Davis	Iowa State University, United States
R. James Duckworth	Worchester Polytechnic Institute, United States
Jose G. Delgado-Frias	Washington State University, United States
Lalinda Fernando	Rockwell Collins, United States
Edward F. Gehringer	North Carolina State University, United States
Ian Greenshields	University of Connecticut, United States
David R. Kaeli	Northeastern University, United States
Ashfaq A. Khokar	University of Delaware, United States
James McDonald	Monmouth University, United States
Rafic Makki	United Arab Emirates University
Hossein S. Moussavinezhad	Western Michigan University, United States
Yale N. Patt	University of Texas-Austin, United States
Hardy J. Pottinger	University of Missouri-Rolla, United States
S.S.S.P. Rao	Indian Institute of Technology, Bombay, India
Albert Reuther	MIT Lincoln Laboratory, United States
Fred U. Rosenberger	Washington University, United States
Kenneth L. Short	SUNY Stony Brook, United States
Adit D. Singh	Auburn University, United States
Mani Soma	University of Washington, United States
J. Carter M. Tiernan	University of Texas at Arlington, United States
A. Yavuz-Oruc	University of Maryland, United States

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* [ACM/IEEECS, 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 2004" or simply CE2004. 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 [ACM/IEEECS, 2001] published in December 2001), information systems ("Computing Curricula: Information Systems" or the IS2002 report [ACM/AIS, 2002] published in 2002), software engineering ("Computing Curricula: Software Engineering" or the SE2004 report [ACM/IEEECS, 2004] published in 2004), 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.

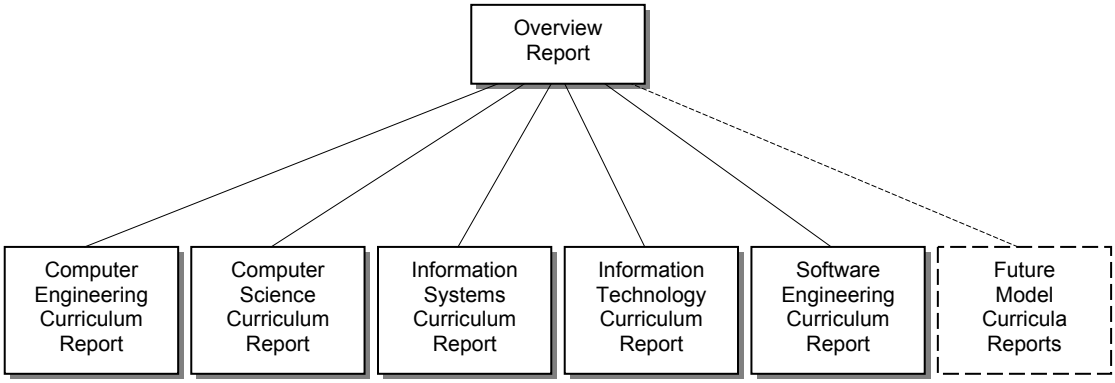

Figure 1.1: Computing curricula reports

1.2 Overview of the CE2004 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 of these was not curricular development, but 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 CE2004 Task Force, the members of which appear at the beginning of this report. Given the scale of the CE2004 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 CE2004 Report

This CE2004 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 outcomes, 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 is defined as the discipline that 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 October 2004, ABET has accredited over 170 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 October 2004

	Year of Initial Accreditation				
Program Name	Before 1980	1980 to 1989	1990 to 1999	2000 to 2004	Totals
Computer Engineering	10	32	44	54	140
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	3	12
Other titles	0	2	1	2	5
Totals	16	46	46	51	173

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 computers and computer-based 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.

Electrical engineering spans a wide range of areas, including bioengineering, power engineering, electronics, telecommunications and digital systems. Related to the field of computer engineering, electrical engineers concern themselves primarily 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 (often 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 support engineers by installing and operating computer-based products, and maintaining 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 in their designs and implementations of 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 should 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 should 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 as part of governmental quality assurance efforts. Each institution is required to demonstrate that their degrees meet the requisite benchmark standards for that discipline. One example of these benchmark standards is [UKQAA, 2000]. This benchmarking defines both threshold (minimal) and modal (average) expectations with respect to demonstrated student knowledge, skills, and judgment. 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.

Many countries have established their own processes for evaluation and/or accreditation through governmental or professional societies. Additionally, for over twenty years ABET has been evaluating programs outside of the United States to determine if they are substantially equivalent to meeting the ABET accreditation criteria [ABET, 2003]. Mutual recognition of the evaluation and/or accreditation process exists through the mechanisms of the Washington Accord [Washington], the Sydney Accord [Sydney], the Dublin Accord [Dublin], FEANI [FEANI], and the International Register of Professional Engineers [IRPE].

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. Graduation from an accredited engineering program is typically a prerequisite step towards professional registration or licensure. 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

COmputer Engineering 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 preparation for 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
CE2004 Discipline Areas Containing Core Material

CE-ALG*	Algorithms
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

* Consult the CC2001 Computer Science report for more detail

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 CE2004 project has been available at [Aub] since 2002.

Table 4.2
CE2004 Presentations

Date	Conference or meeting	Type
2002 June 16-19	American Society for Engineering Education – Montreal, Canada [ASEE'02]	Panel
2002 November 6-9	Frontiers in Education – Boston, USA [FIE'02]	Panel
2003 February 19-23	SIGCSE Technical Symposium – Reno, USA [SIGCSE'03]	Panel
2003 March	Electrical and Computer Engineering Department Heads Association – Hawaii, USA	Panel
2003 June 22-25	American Society for Engineering Education – Nashville, USA [ASEE'03]	Panel
2003 June 29 – July 2	Innovation and Technology in Computer Science Education – Thessaloniki, Greece [ITiCSE'03]	Poster
2003 November 5-8	Frontiers in Education - Denver , USA [FIE'03]	Panel &Paper
2004 March 3-7	SIGCSE Technical Symposium – Norfolk, USA [SIGCSE'04]	Panel
2004 June 20-23	American Society for Engineering Education – Salt Lake City, USA [ASEE'04]	Paper
2004 October 20-23	Frontiers in Education – Savannah, USA [FIE'04]	Panel & Paper
2005 February	SIGCSE Technical Symposium – St. Louis, USA [SIGCSE'05]	BOAF

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. For example, CE-CAO3 contains nine topics, such as "Virtual Memory Systems".

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 two to three times the in-class time. Thus, a unit that is listed as requiring three hours will typically entail a total of nine to twelve hours (three in-class hours and six to 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 outcomes for that unit. It may be appropriate to spend more time on a knowledge unit than the mandated minimum.

- *The 420 core hours specified do not include time for laboratories, design, math, science, etc.* These activities and subjects should be added to the 420 core hours as necessary to provide supporting material and preparation for engineering practice.

The number of hours shown should be sufficient to ensure familiarity with the topics, but not necessarily to achieve expertise in that area. The number of core hours was deliberately kept to a minimum to allow programs the flexibility to emphasize selected areas in accordance with the specific objectives, prerequisite structure, and level of student preparation in that program. Therefore, the actual time devoted to a particular core topic will vary from program to program, with some programs spending more than the specified minimum number of hours on selected core topics, while devoting only the minimum level of coverage to others.

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.

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 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 are 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 one year of mathematics and science, one year of computer engineering core, one-half year of computer engineering electives, one-half year of additional engineering studies, and one 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 can 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.

Table 4.3
The Computer Engineering Body of Knowledge

Computer Engineering Knowledge Areas and Units	
CE-ALG Algorithms [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

* Consult the CC2001 Report [ACM/IEEECS, 2001] for more detail on these knowledge units

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

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. 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. The following sections provide guidance on how design may be incorporated within the computer engineering curriculum.

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 safety-critical 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 may 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

In today's world there are pressures on institutions to ensure that graduates have the capacity to meet the needs of employers. A more positive view is that institutions can be agents of change, producing graduates who are capable of moving into employment with skills and expectations that benefit their employers.

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 arises 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 Student Learning and Assessment

Student learning is very complex and this is not the venue to discuss the ramifications of that process. However, some basic elements of learning are of interest. Students tend to learn in stages as described by Grow [Grow 1991] starting with directed learning and ending with self-directed learning. Grow showed (Table 5.1) the learning stages in which learning is possible and the manner of participation by student and teacher. Teachers should be aware of these stages to assure that students receive the proper education as they progress through these stages.

Table 5.1
Learning Stages [Grow 1991]

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

One should observe a number of important considerations in the assessment 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.

5.10 Lifelong Learning

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 abreast with changes, 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 current, the equipment has to be up-to-date, and faculty members need to be engaged in relevant scholarship. 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 current and interesting information.

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.

5.11 The Business Perspective

To complement the technical side of their experiences, computer engineers need 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

profitable business ventures, both within established companies and in entrepreneurial endeavors. 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.

5.12 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
- 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 need 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

Social context 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 social 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

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, 1990]
- Association for Computing Machinery (ACM): *ACM Code of Ethics and Professional Conduct* [ACM, 1992]
- 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).

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 introduced in Chapter 4, and presented in Appendix A, 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 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 continue. 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) 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

As described in Chapter 6, 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.

In Chapter 5, 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 issues can have the highly undesirable effect of stifling beneficial innovation. Teachers should 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, as described in Chapter 5. 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 within 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.

7.3 Course Material Presented by Other Departments

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 appears in courses outside of the department in which computer engineering resides. In some cases, students may have learned this material prior to entering the computer engineering program.

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 an essential component of computer engineering. 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 knowledge of the mathematical principles of discrete structures and exposure to related tools. 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, as specified in the computer engineering body of knowledge.
- *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, and artificial intelligence. 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 represents a vital component of logical thought within the field of computer engineering. The scientific method (hypothesis formation, experimentation and data collection, analysis) 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 knowledge of basic sciences, such as physics and chemistry. Basic physics concepts in electricity and magnetism form the basis for much of the underlying electrical engineering content in the body of knowledge. Other science courses, 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 Other Requirements

Many institutions have other requirements that apply to all students, such as general education requirements. The size and content of this requirement varies widely, depending on the home country, the institutional mission, legal requirements, and other factors. Such 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 and intermediate phases of the curriculum. 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. While basic material in many of these areas is already included in the core, these electives 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. This list is by no means exhaustive. 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, computer science and engineering or computer and systems 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. 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 42 hours for instruction.

- Students should experience at least two computer engineering modules in the first year of study, at least four in the second year of study, and at least five 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 affect curriculum design. For schools with existing engineering programs, much of what follows may already be in place or understood.

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.

Finally, with the availability of up-to-date reference materials on the World Wide Web, access to such resources as the IEEE Digital Library and the ACM Digital Library should be provided.

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.

Endnote References to this Report

[ABET, 2003] Policies and procedures for ABET substantial equivalency evaluations, <http://www.abet.org/international/policies.html>, 2003.
[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/AIS, 2002] IS2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems, Association for Computing Machinery (ACM), Association for Information Systems (AIS), Association for Information Technology Professionals (AITP), 2002.
[ACM/IEEECS, 1991] Computing Curricula 1991, Report of the ACM/IEEE-CS Joint Curriculum Task Force, IEEE Computer Society Press [ISBN 0-8186-2220-2] and ACM Press [ISBN 0-8979-381-7], 1991.
[ACM/IEEECS, 1999] CAN and IEEE Computer Society, Software Engineering Code of Ethics and Professional Practice, <http://computer.org/certification/ethics.htm>, 1999.
[ACM/IEEECS, 2001] Computing Curriculum 2001, Computer Science, IEEE Computer Society Press and ACM Press, December 15, 2001.
[ACM/IEEECS, 2004] Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, IEEE Computer Society Press and ACM Press, August 23, 2004.
[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.
[Dublin] Dublin Accord, http://www.engc.org.uk/international/dublin.asp.
[EAB, 1986] Educational Activities Board. Design education in computer science and engineering. Technical Report 971, Computer Society of the IEEE, October 1986.
[FEANI] FEANI-European Federation of National Engineering Associations, http://www.feani.org.
[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.
[Grow 1991] Grow, G.O. "Teaching Learners to be Self-Directed", Adult Education Quarterly, Vol. 41(3), pp. 125-149. 1991.
[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.
[IRPE] International Register of Professional Engineers, http://www.engc.org.uk/international/irpe.asp.
[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
[Langdon, et. al. 1986] Design Education in Computer Science and Engineering, Technical Report, IEEE Computer Society Educational Activities Board, October 1, 1986.
[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.
[SIGCSE'05] SIGCSE Technical Symposium, <http://www.csc.vill.edu/sigcse2004/>, St. Louis, Missouri, 23-27 February 2005.
[Sydney] Sydney Accord, http://www.engc.ork.uk/international/sydney.asp.
[UKQAA, 2000] Quality Assurance Agency for Higher Education, "Computing, a report on benchmark levels for Computing," Southgate House, Gloucester, England, April 2000.
[Washington] Washington Accord, http://www.washingtonaccord.org/Default.htm.

Bibliography

[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, 2003] Policies and procedures for ABET substantial equivalency evaluations, <http://www.abet.org/international/policies.html>, 2003.

[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/AIS, 2002] IS2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems, Association for Computing Machinery (ACM), Association for Information Systems (AIS), Association for Information Technology Professionals (AITP), 2002.

[ACM/IEEECS, 1991] Computing Curricula 1991, Report of the ACM/IEEE-CS Joint Curriculum Task Force, IEEE Computer Society Press [ISBN 0-8186-2220-2] and ACM Press [ISBN 0-8979-381-7], 1991.

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

[ACM/IEEECS, 2001] Computing Curriculum 2001, Computer Science, IEEE Computer Society Press and ACM Press, December 15, 2001.

[ACM/IEEECS, 2004] Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, IEEE Computer Society Press and ACM Press, August 23, 2004.

[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.

[Dublin] Dublin Accord, http://www.engc.org.uk/international/dublin.asp.

[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.

[FEANI] FEANI-European Federation of National Engineering Associations, http://www.feani.org.

[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. THIS NEEDS TO CHANGE

[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.

[Grow 1991] Grow, G.O. "Teaching Learners to be Self-Directed", Adult Education Quarterly, Vol. 41(3), pp. 125-149. 1991.

[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.

[IRPE] International Register of Professional Engineers, http://www.engc.org.uk/international/irpe.asp.

[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.
[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.

[SIGCSE'05] SIGCSE Technical Symposium, <http://www.csc.vill.edu/sigcse2004/>, St. Louis, Missouri, 23-27 February 2005.

[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/.

[Sydney] Sydey Accord, http://www.engc.ork.uk/international/sydney.asp.

[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.

[Washington] Washington Accord, http://www.washingtonaccord.org/Default.htm.

[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.

Appendix A

The Computer Engineering Body of Knowledge

This appendix to the *Computing Curricula - Computer Engineering (CE2004)* report defines the knowledge domain that is likely to appear in an undergraduate curriculum in computer engineering. The underlying rationale for this categorization scheme and additional details about its history, structure, and application are included in the body of the report. Included with this appendix is a summary of the fundamental concepts that are necessary to understand the recommendations.

A.1 Introduction

The Task Force developed this model curriculum by first defining the primary disciplines that make up the body of knowledge for computer engineering. Some of these areas contain material that should be part of *all* computer engineering curricula. However, other areas contain material that might or might not be part of such curricula, depending on the specific educational objectives of a program. The areas that contain material that should be included in all computer engineering curricula are:

CE-ALG*	Algorithms
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-DSC*	Discrete Structures
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-PRS	Probability and Statistics
CE-SPR*	Social and Professional Issues
CE-SWE*	Software Engineering
CE-VLS	VLSI Design and Fabrication

* Consult the CC2001 Computer Science Report for more detail

A.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.

A.3 Core and Elective Units

One of the basic goals was to keep the required component of the body of knowledge as small as possible. To implement this principle, a minimal **core** has been defined, comprising those units for which there is broad consensus that the corresponding material is essential to anyone obtaining an undergraduate degree in computer engineering. Units taught as part of an undergraduate program that fall outside the core, are *elective* to the curriculum, even though some programs may choose to require them.

The following points are emphasized:

- *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.

A.4 Time Required to Cover a Knowledge Unit

For consistency with the Computer Science Report and earlier curriculum reports, we have 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 the Task Force has used a metric with its roots in a classical lecture-oriented form, we believe 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 reparation time of the time students spend outside of class. As a general guideline, the amount of out-of-class work for a student is approximately two to three times the in-class time. Thus, a unit that requires three hours of instruction typically entails a total of nine to twelve hours (three in class and six to nine outside).

The hours listed for a unit represent a minimum level of coverage. We should interpret the time measurements assigned for each unit as the minimum amount of time necessary to enable a student to perform the learning outcomes for that unit. It is always appropriate to spend more time on a unit than the suggested minimum.

The 420 core hours specified do not include time for laboratories, design, math, science, etc. These activities and subjects should be added to the 420 core hours as necessary to provide supporting material and preparation for engineering practice.

The number of core hours was deliberately kept to a minimum to allow programs the flexibility to emphasize selected areas in accordance with the specific objectives, prerequisite structure, and level of student preparation in that program. Therefore, the actual time devoted to a particular core topic will vary from program to program, with some programs spending more than the specified minimum number of hours on selected core topics, while devoting only the minimum level of coverage to others.

A.5 Summary of the Computer Engineering Body of Knowledge

A summary of the Body of Knowledge—showing the areas, units, which units are core, and the minimum time required for each—appears as Table A-1. The details of each section of the body of knowledge follow as separate sections.

Table A-1 The Computer Engineering Body of Knowledge

Computer Engineering Knowledge Areas and Units	
CE-ALG Algorithms [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

* Consult the CC2001 Report [ACM/IEEECS, 2001] for more detail on these knowledge units

A.6 Comments on Knowledge Areas

The following sections provide comments on the individual areas in the computer engineering body of knowledge. They appear here to help the reader understand how these areas contribute to the overall computer engineering curriculum.

A.6.1 Comments on Algorithms

Algorithms are fundamental to computer engineering. The real-world performance of any software or hardware system depends on two things: (1) the algorithms chosen, and (2) the suitability and efficiency of the implementation. Good algorithm design is, therefore, crucial for the performance of all systems. Moreover, the study of algorithms provides insight into the intrinsic nature of the problem as well as possible solution techniques independent of programming language, computer hardware, or any other implementation aspect.

An important part of computing is the ability to select algorithms appropriate to particular purposes and to apply them, recognizing both the likelihood that multiple reasonable solutions exist and the possibility that no suitable algorithm may exist. This facility relies on understanding the range of algorithms that address an important set of well-defined problems, recognizing their strengths and weaknesses, and their suitability in particular contexts. Efficiency is a pervasive theme throughout this area.

A.6.2 Comments on Computer Architecture and Organization

Computer architecture is a key component of computer engineering and the practicing computer engineer should have a practical understanding of this topic. It is concerned with all aspects of the design and organization of the central processing unit and the integration of the CPU into the computer system itself. Architecture extends upward into computer software because a processor's architecture must cooperate with the operating system and system software. It is difficult to design an operating system well without knowledge of the underlying architecture. Moreover, the computer designer must have an understanding of software in order to implement the optimum architecture.

The computer architecture curriculum has to achieve multiple objectives. It must provide an overview of computer architecture and teach students the operation of a typical computing machine. It must cover basic principles, while acknowledging the complexity of existing commercial systems. Ideally, it should reinforce topics that are common to other areas of computer engineering; for example, teaching register indirect addressing reinforces the concept of pointers in C. Finally, students must understand how various peripheral devices interact with, and how they are interfaced to a CPU.

A.6.3 Comments on Computer Systems Engineering

Computer engineers build systems containing hardware and software components, usually as part of a larger system. Included is the development of new devices such as digital camera, hand-held computers, location aware systems, etc. Embedded computer systems developments are becoming pervasive.

We must make decisions regarding the way to design to have maximum impact and effect at the system level. Decisions have to be made about alternative approaches, trade-offs need to be addressed, and decisions on all these matters need to be justified on grounds of technical insight and judgment. Often the computer engineer will be part of a multi-disciplinary team and will have to react accordingly.

A.6.4 Comments on Circuits and Signals

Circuits and signals are foundational material for computer engineering. These areas provide the basic knowledge for the design of the circuits used to implement computers. A knowledge of the electrical circuits used to implement digital circuits and computers should include basic electrical quantities, resistive and reactive circuits, sinusoidal analysis, convolution, and frequency selective circuits. This is a very broad area and one should expect a great deal of variation between programs for the coverage of topics outside of the core.

A.6.5 Comments on Database Systems

Typically, users of computers have to deal with massive amounts of information on a daily basis; there are e-mails, documents, records, addresses, web sites and many other kinds of information. In the context of technical development, there can be specifications, designs, tests, implementations, different tools and different versions of these tools; all of these can relate to hardware, software, communications, and so on.

Database systems are designed to maintain and manage large collections of information, including relationships between elements and access to data. The computer engineering student needs to be able to develop conceptual and physical data models, determine what methods and techniques are appropriate for a given problem, and be able to select and implement an appropriate solution that reflects all suitable constraints, including scalability and usability.

A.6.6 Comments on Digital Logic

The logic design area covers the digital building blocks, tools, and techniques in the design of computers and other digital systems. Emphasis is on a building-block approach. Extensive core material is included in this area as digital logic design is one of the topic areas that differentiate computer engineers from electrical engineers and computer scientists. This core material covers a variety of basic topics, including switching theory, combinational and sequential logic circuits, and memory elements.

Topics of a more advanced nature include design with programmable logic and field-programmable gate arrays (FPGAs), modeling and simulation, digital system design, verification, and fault models and testing.

A.6.7 Comments on Discrete Structures

The area of discrete structures is foundational material for computer engineering, including important material from such areas as set theory, logic, methods of proofs, graph theory, combinatorics, and recursion. The material in discrete structures is pervasive in the areas of data structures and algorithms. As the field of computer engineering matures, more and more sophisticated analysis techniques affect practical problems. To understand the computational techniques of the future, today's students will need a strong background in discrete structures.

It is important to remember that the study of discrete structures must occur in the context of computer engineering. Wherever possible, reference should include engineering situations or settings and the topics in discrete structures should integrate with themes from computer engineering. It is important to emphasize application rather than just theory.

Finally, we note that while areas often have somewhat fuzzy boundaries, this is especially true for discrete structures. We have assembled a body of material of a mathematical nature that must be included in a computer engineering education and that computer engineering educators know well enough to specify it in detail. However, the decision about where to draw the line between particular knowledge areas on the one hand, and topics left only as supporting mathematics on the other, was inevitably somewhat arbitrary.

A.6.8 Comments on Digital Signal Processing

Digital signal processing can be applied to the transformation, synthesis and analysis of data. For example, when modeling a communication channel, filters, generators and analyzers can be used to remove, add or measure noise in processing audio, images and video. Digital signal processing can also involve domain-specific symbolic processing, which is typically named for the type of data used for input and output. For example, if we input numerical data and output symbolic data, we call the field *pattern recognition*. If we input voice and output text, we call it *voice recognition.* If we input images and output symbols, we call it *computer* or *machine vision*. If we input text and output voice, we call it *voice synthesis*. Using the broadest interpretation of the digital signal processing term, any of these areas could be included when selecting courses that support the digital signal-processing domain.

Most broadly speaking, the kind of numerical digital signal processing performed is a function of the dimensionality of the data. In one-dimension, a signal can be generated by any single-valued numerical function or digitized from any time-varying form of energy. Examples include pressure waves (voice, audio, etc.), sensor measurements (temperature, range-to-target, speed), and sensor fusion (i.e., mechanisms to estimate the state of a plant given a model and multiple sensors). In two-dimensions, digital signal processing is a kind of numerical data processing that deals with images (typically called *image processing)*. In three dimensions, digital signal processing is sometimes called *image sequence processing* or *video processing*.

Digital signal processing is a broad area. It is expected that variation will exist among programs outside of the core.

A.6.9 Comments on Electronics

Electronics is foundational material for computer engineering. These areas provide the basic knowledge for the design of the electronic circuits used to implement computers. Basic core material includes the electronic properties of materials, diodes, logic families and storage elements. More advanced topics include design parameters, interfacing and buses, circuit modeling and simulation, and operational amplifiers. This is a very broad area and it is expected that there will be a great deal of variation between programs in the coverage of topics outside of the core.

A.6.10 Comments on Embedded Systems

Almost every electronic appliance and device today uses embedded systems. Cell phones, automobiles, toasters, televisions, airplanes, medical equipment, and a host of other devices, products, and applications use embedded systems. Such systems include microcontrollers, embedded programs, and real-time operating systems. These systems requires a conscious effort to produce the most reliable product possible requiring the utmost diligence in system design and in design methodologies. Indeed, these designs often reflect the design of low power systems and tool support.

A.6.11 Comments on Human-Computer Interaction

The design and development of displays, alarms, and interfaces for small or large screens (some of which may involve interaction) is an activity captured in the study of the human computer interface and in a study of human computer interaction. This discipline is increasingly software based and design needs require guidance by insights from psychology and informed by an appreciation of human diversity including matters such as colored blindness and deafness; in these circumstances, multimedia approaches often have a role to play. It is important to note that in certain applications there are crucial requirements for reliability and other kinds of performance that have implications for matters such as safety and security.

Emphasis is placed on understanding human reactions to displays of various kinds and on human behavior in the context of interactive objects. Based on these, students need to understand the principles associated with the evaluation of interfaces including those that embody interaction. Students need to

know the principles and guidelines that reflect best practice in the design, development, and maintenance of interfaces for multiple types of systems.

A.6.12 Comments on Computer Networks

The number of computer networks is increasing dramatically. From small offices to entire countries, computer networks have become the heart of electronic communication today. Using established protocols, these local and wide area networks have become the conduit for servers and clients. Of interest today is data integrity and security as well as the "right" to the information communicated. With wireless and mobile computing, it has become even more essential that companies and governments preserve the integrity of such communication vehicles. Increasingly, the use of data compression has helped the efficiency of data communications, where the stress on performance is an increasing concern.

A.6.13 Comments on Operating Systems

An operating system defines a software interface of the computer hardware and the architecture with which computer engineers can control and exploit the hardware to provide maximum benefit to the user. It also manages sharing of resources (hardware and software) among the computer's users (user programs and systems programs).

Student should understand the basic principles and the purposes of an operating system prior to a study of digital instrumentation and embedded systems. It is necessary to addresses both the use of operating systems (externals) and their design and implementation (internals). Many of the ideas involved in operating system use have wider applicability across the field of computer engineering such as concurrent programming. Studying internal design has relevance in such diverse areas as fault tolerance, algorithm design and implementation, modern device development, building virtual environments, building secure and safe systems, network management, and many other areas.

A.6.14 Comments on Programming Fundamentals

Competency in a programming language is prerequisite to the study of computer engineering. Undergraduate programs must teach students how to use at least one programming language. The difficulty of achieving the necessary level of fluency in a programming language is further complicated by the need to include many advanced techniques. Students should cover the core topics in this unit to receive exposure to the basic pieces that should be covered independent of a particular programming language as programming languages tend to come and go over the years.

Object-oriented programming, event-driven applications, and the use of extensive APIs (application programming interfaces) have become fundamental tools that some computer engineering students need early in their academic program. These concepts may be included in a program that only teaches an object-oriented language such as C++ or Java.

A.6.15 Comments on Probability and Statistics

The topics of probability and statistics provide important insights into a range of topics of fundamental importance to the computer engineer. For example, all issues of reliability and dependability rely on an understanding of these topics. However, additionally, they play fundamental roles in testing and evaluation (of hardware, software, and communications systems) where one must guarantee levels of performance. Further uses of the topics are available in a wide variety of areas: searching, data structure design and implementation (hash tables), computer architecture (cache concerns), operating systems (working set models), human computer interaction (experimentation), security and in aspects of intelligent systems.

A.6.16 Comments on Social and Professional Issues

Students must develop an understanding of the social and professional context in which they apply their computer engineering education. Ethical considerations must be covered in context of technical topics, otherwise it will reinforce the false notion that technological processes are void of ethical issues. Thus, it is important that several courses include modules that analyze ethical considerations in the context of the technical subject matter.

Ethics-related modules may occur in almost any course in the curriculum. Courses in areas such as software engineering, databases, computer networks, data mining, and human computer interfaces provide obvious context for analysis of ethical issues and should arise naturally from those subjects. For example, a programming assignment built around applications such as controlling the movement of a laser during eye surgery can help to address the professional, ethical, and social impacts of computing.

Computer engineers must be cognizant of their responsibility to the public. They must also be aware of the potential conflicts between the obligations to their employer and the obligations to the customer, user, and others affected by their work.

A.6.17 Comments on Software Engineering

Software engineering is the discipline concerned with the application of theory, knowledge, and practice to build effectively and efficiently software systems that satisfy the requirements of users and customers. Software engineering is applicable to small, medium, and large-scale systems. It encompasses all phases of the life cycle of a software system. The life cycle includes requirement analysis and specification, design, construction, testing, and operation and maintenance. The development of programs benefits from the concepts and practices derived from software engineering. There is a need to introduce fundamental ideas from software engineering into elementary programming and into early experience of software design.

Software engineering employs engineering methods, processes, techniques, and measurement. It benefits from the use of tools for managing software development; analyzing and modeling software artifacts; assessing and controlling quality; and for ensuring a disciplined, controlled approach to software evolution and reuse. Software development, which can involve an individual developer or a team of developers, requires choosing the tools, methods, and approaches that are most applicable for a given development environment.

A.6.18 Comments on VLSI Design and Fabrication

The design of the integrated circuits used to implement computers and associated hardware contains some core material. This core includes basic properties of materials, the structure of inverters, combinational and sequential logic structures, and memories and logic arrays. This is a very broad area and it is expected that there will be a great deal of variation between programs in the coverage of topics outside the core.

A.7 Details of the Body of Knowledge

The following pages present the 18 areas that comprise the computer engineering body of knowledge. Each includes a list of the knowledge units within that area, a detailed list of topics and learning outcomes for each unit, and for core units, a recommended minimum coverage time.

Algorithms (CE-ALG)

CE-ALG0 History and overview [core]
CE-ALG1 Basic algorithmic analysis [core] *
CE-ALG2 Algorithmic strategies [core] *
CE-ALG3 Computing algorithms [core] *
CE-ALG4 Distributed algorithms [core] *
CE-ALG5 Algorithmic complexity [core] *
CE-ALG6 Basic computability theory [elective] *

* Consult the CC2001 Report [ACM/IEEECS, 2001] AL Knowledge Area for more detail

CE-ALG0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying analysis, complexity, and algorithmic strategies
- Highlight some people that contributed or influenced the area of algorithms
- Mention some basic algorithms and some reasons for their differences
- Highlight how the use of theory influences algorithms
- Indicate how algorithms are part of many different computer applications
- Provide some knowledge themes such as relating complexity with algorithms
- Contrast complexities of different algorithmic strategies
- Explore some additional resources associated with algorithms
- Explain the purpose and role of algorithms in computer engineering

Learning outcomes:
1. Identify some contributors to algorithms and and relate their achievements to the knowledge area.
2. Associate some of the themes involved with algorithms.
3. Name some applications where algorithms are important.
4. Relate contributors with their achievements to the subject.
5. Describe how computer engineering uses or benefits from algorithms.

CE-ALG1 Basic algorithmic analysis [core]
Minimum core coverage time: 4 hours

Topics:
- Asymptotic analysis of upper and average complexity bounds
- Identifying differences among best, average, and worst case behaviors
- Big "O," little "o," omega, and theta notation
- Empirical measurements of performance
- Time and space tradeoffs in algorithms
- Using recurrence relations to analyze recursive algorithms

Learning outcomes:
1. Use big O, omega, and theta notation to give asymptotic upper, lower, and tight bounds on time and space complexity of algorithms.
2. Determine the time complexity of simple algorithms.
3. Deduce the recurrence relations that describe the time complexity of recursively-defined algorithms, and solve simple recurrence relations.

CE-ALG2 Algorithmic strategies [core]
Minimum core coverage time: 8 hours

Topics:
- Brute-force/exhaustive search algorithms
- Greedy algorithms
- Divide-and-conquer
- At least one of: Backtracking, branch-and-bound, heuristics

Learning outcomes:
1. Design algorithms using the brute-force, greedy, and divide-and-conquer strategies.
2. Design an algorithm using at least one other algorithmic strategy from the list of topics for this unit.

CE-ALG3 Computing algorithms [core]
Minimum core coverage time: 12 hours

Topics:
- Simple numerical algorithms
- Sequential and binary search algorithms
- Sorting algorithms
- Hash tables, including collision-avoidance strategies
- Binary search trees
- Representations of graphs (adjacency list, adjacency matrix)
- Depth- and breadth-first traversals
- Shortest-path algorithms (Dijkstra's and Floyd's algorithms)
- Transitive closure (Floyd's algorithm)
- Minimum spanning tree (Prim's and Kruskal's algorithms)
- Topological sort

Learning outcomes:
1. Use and implement the fundamental abstract data types—specifically including hash tables, binary search trees, and graphs—necessary to solve algorithmic problems efficiently.
2. Solve problems using efficient sorting algorithms, and fundamental graph algorithms, including depth-first and breadth-first search, single-source and all-pairs shortest paths, transitive closure, topological sort, and at least one minimum spanning tree algorithm.
3. Demonstrate the following abilities: to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in simple programming contexts.

CE-ALG4 Distributed algorithms [core]
Minimum core coverage time: 3 hours

Topics:
- Concurrency
- Scheduling
- Fault tolerance

Learning outcomes:
1. Explain the distributed paradigm.
2. Distinguish between logical and physical clocks.
3. Describe the relative ordering of events.
4. Explain one simple distributed algorithm.

CE-ALG5 Algorithmic complexity [core]
Minimum core coverage time: 2 hours

Topics:
- Tractable and intractable problems
- Definition of the classes P and NP
- NP-completeness (Cook's theorem)
- Standard NP-complete problems
- Uncomputable functions
- The halting problem
- Implications of uncomputability

Learning outcomes:
1. Define the classes P and NP.
2. Explain the significance of NP-completeness.
3. Prove that a problem is NP-complete by reducing a classic known NP-complete problem to it.

CE-ALG6 Basic computability theory [elective]

Topics:
- Deterministic finite Automata (DFA)
- Non-deterministic finite Automata (NFA)
- Equivalence of DFA's and NFA's
- Context-free grammars
- Pushdown automata (PDA)

Learning outcomes:
1. Explain the idea that some problems may have no algorithmic solution.
2. Provide examples that illustrate the implications of uncomputability.

Computer Architecture and Organization (CE-CAO)

CE-CAO0 History and overview [core]
CE-CAO1 Fundamentals of computer architecture [core]
CE-CAO2 Computer arithmetic [core]
CE-CAO3 Memory system organization and architecture [core]
CE-CAO4 Interfacing and communication [core]
CE-CAO5 Device subsystems [core]
CE-CAO6 Processor systems design [core]
CE-CAO7 Organization of the CPU [core]
CE-CAO8 Performance [core]
CE-CAO9 Distributed system models [core]
CE-CAO10 Performance enhancements [elective]

CE-CAO0 History and overview of computer architecture and organization [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying computer architecture and organization
- Highlight some people that influenced or contributed to the area of computer architecture and organization
- Indicate some important topic areas such as system organization and architecture, memory, interfacing, microprocessors, and performance
- Contrast the meanings of between computer organization and computer architecture
- Indicate the importance of doing binary arithmetic with computers
- Mention memory as a crucial component to the design of a computer
- Illustrate the importance of interfacing with computer components and peripherals
- Mention a typical CPU and sketch its organization
- Indicate why performance leads to alternate architectures
- Mention caching a way to improve performance
- Mention some of the strategies used in architecture such as CISC and RISC approaches
- Mention the strategies of multiprocessing strategies and their potential
- Explore some additional resources associated with computer architecture and organization
- Explain the purpose and role of computer architecture and organization in computer engineering

Learning outcomes:
1. Identify some contributors to computer architecture and organization and relate their achievements to the knowledge area.
2. Explain the reasons and strategies for different architectures.
3. Articulate differences between computer organization and computer architecture.
4. Identify some of the components of a computer.
5. Indicate some strengths and weaknesses inherent in different architectures.
6. Describe how computer engineering uses or benefits from computer architecture and organization.

CE-CAO1 Fundamentals of computer architecture [core]
Minimum core coverage time: 10 hours

Topics:
- Organization of the von Neumann machine
- Instruction formats
- The fetch/execute cycle; instruction decoding and execution
- Registers and register files
- Instruction types and addressing modes
- Subroutine call and return mechanisms
- Programming in assembly language
- I/O techniques and interrupts
- Other design issues

Learning outcomes:
1. Explain the organization of a von Neumann machine and its major functional units.
2. Explain how a computer fetches from memory and executes an instruction.
3. Articulate the strengths and weaknesses of the von Neumann architecture.
4. Explain the relationship between the representation of machine level operation at the binary level and their representation by a symbolic assembler.
5. Explain why a designer adopted a given different instruction formats, such as the number of addresses per instruction and variable length vs. fixed length formats.
6. Write small programs and fragments of assembly language code to demonstrate an understanding of machine level operations.
7. Implement some fundamental high-level programming constructs at the machine-language level.
8. Use computer simulation packages to investigate assembly language programming.

CE-CAO2 Computer arithmetic [core]
Minimum core coverage time: 3 hours

Topics:
- Representation of integers (positive and negative numbers)
- Algorithms for common arithmetic operations (addition, subtraction, multiplication, division)
- Significance of range, precision, and accuracy in computer arithmetic
- Representation of real numbers (standards for floating-point arithmetic)
- Algorithms for carrying out common floating-point operations
- Converting between integer and real numbers
- Multi-precision arithmetic
- Hardware and software implementation of arithmetic unit
- The generation of higher order functions from square roots to transcendental functions

Learning outcomes:
1. Appreciate how numerical values are represented in digital computers
2. Understand the limitations of computer arithmetic and the effects of errors on calculations.
3. Appreciate the effect of a processor's arithmetic unit on its overall performance,

CE-CAO3 Memory system organization and architecture [core]
Minimum core coverage time: 8 hours

Topics:
- Memory systems hierarchy
- Coding, data compression, and data integrity
- Electronic, magnetic and optical technologies
- Main memory organization and its characteristics and performance
- Latency, cycle time, bandwidth, and interleaving
- Cache memories (address mapping, line size, replacement and write-back policies)
- Virtual memory systems
- Memory technologies such as DRAM, EPROM, and FLASH
- Reliability of memory systems; error detecting and error correcting systems

Learning outcomes:
1. Identify the main types of memory technology.
2. Explain the effect of memory latency and bandwidth on performance.
3. Explain the use of memory hierarchy to reduce the effective memory latency.
4. Describe the principles of memory management.
5. Understand how errors in memory systems arise and how to resolve them.

CE-CAO4 Interfacing and communication [core]
Minimum core coverage time: 10 hours

Topics:
- I/O fundamentals: handshaking, buffering,
- I/O techniques: programmed I/O, interrupt-driven I/O, DMA
- Interrupt structures: vectored and prioritized, interrupt overhead, interrupts and reentrant code
- Memory system design and interfacing
- Buses: bus protocols, local and geographic arbitration

Learning outcomes:
1. Explain how to use interrupts to implement I/O control and data transfers.
2. Write small interrupt service routines and I/O drivers using assembly language.
3. Identify various types of buses in a computer system.
4. Describe data access from a magnetic disk drive.
5. Analyze and implement interfaces.

CE-CAO5 Device subsystems [core]
Minimum core coverage time: 5 hours

Topics:
- External storage systems; organization and structure of disk drives and optical memory
- Basic I/O controllers such as a keyboard and a mouse
- RAID architectures
- Video control
- I/O Performance
- SMART technology and fault detection
- Processor to network interfaces

Learning outcomes:
1. Compute the various parameters of performance for standard I/O types.
2. Explain the basic nature human computer interaction devices.
3. Describe data access from magnetic and optical disk drives.

CE-CAO6 Processor systems design [core]
Minimum core coverage time: 10 hours

Topics:
- The CPU interface: clock, control, data and address buses
- Address decoding and memory interfacing
- Basic parallel and serial interfaces
- Timers
- System firmware

Learning outcomes:
1. Understand how a CPU chip becomes a complete system.
2. Design an interface to memory
3. Understand how to interface and use peripheral chips
4. Write sufficient EPROM-based system software to create a basic stand-alone system.
5. Specify and design simple computer interfaces.

CE-CAO7 Organization of the CPU [core]
Minimum core coverage time: 10 hours

Topics:
- Implementation of the von Neumann machine
- Single vs. multiple bus datapaths
- Instruction set architecture; machine architecture as a framework for encapsulating design decisions
- Relationship between the architecture and the compiler
- Implementing instructions
- Control unit: hardwired realization vs. microprogrammed realization
- Arithmetic units, for multiplication and division
- Instruction pipelining
- Trends in computer architecture: CISC, RISC, VLIW
- Introduction to instruction-level parallelism (ILP)
- Pipeline hazards: structural, data and control
- Reducing the effects of hazards

Learning outcomes:
1. Compare alternative implementation of datapaths.
2. Discuss the generation of control signals using hardwired or microprogrammed implementations.
3. Explain basic instruction level parallelism using pipelining and the major hazards that may occur.
4. Explain what has been done to overcome the effect of branches
5. Discuss the way in which instruction sets have evolved to improve performance; for example, predicated execution.

CE-CAO8 Performance [core]
Minimum core coverage time: 3 hours

Topics:
- Metrics for computer performance; clock rate, MIPS, Cycles per instruction, benchmarks
- Strengths and weaknesses of performance metrics
- Averaging metrics: arithmetic, geometric and harmonic
- The role of Amdahl's law in computer performance

Learning outcomes:
1. Understand the factors that contribute to computer performance.
2. Understand the limitations of performance metrics
3. Select the most appropriate performance metric when evaluating a computer.
4. Discuss the impact on control and datapath design for performance enhancements.

CE-CAO9 Distributed system models [core]
Minimum core coverage time: 3 hours

Topics:
- Classification of models: parallel machine models (SIMD, MIMD, SISD, MISD): Flynn's taxonomy, Handler's classification, message passing
- Granularity, levels of parallelism
- Multiprocessors and multi-computers: Topology, tightly coupled and loosely coupled architectures
- Processes: threads, clients, servers, code migration, software agents
- Physical and logical clocks: clock synchronizing algorithms, Lamport timestamps, vector timestamps
- Election Algorithms
- Mutual Exclusion algorithms
- Distributed transactions: models, classification, concurrency control

Learning outcomes:
1. Explain the differences between different paradigms and their usefulness and applicability
2. Understand how client server model works in a decentralized fashion
3. Understand how agents work and how they solve simple tasks.
4. Understand the concept of logical clocks vs. physical clocks and how they affect implementation of distributed systems
5. Be familiar with simple election and mutual exclusion algorithms and their applicability.

CE-CAO10 Performance enhancements [elective]

Topics:
- Superscalar architecture
- Branch prediction
- Prefetching
- Speculative execution
- Multithreading
- Scalability
- Short vector instruction sets: Streaming extensions, AltiVec, relationship between computer architecture and multimedia applications

Learning outcomes:
1. Discuss how various architectural enhancements affect system performance.
2. Discuss how to apply parallel processing approaches to design scalar and superscalar processors.
3. Discuss how to apply vector-processing techniques to enhance instruction sets for multimedia and signal processing.
4. Understand how each of the functional parts of a computer system affects its overall performance.
5. Estimate the effect on system performance of changes to functional units.

Computer Systems Engineering (CE-CSE)

CE-CSE0 History and overview [core]
CE-CSE1 Life cycle [core]
CE-CSE2 Requirements analysis and elicitation [core]
CE-CSE3 Specification [core]
CE-CSE4 Architectural design [core]
CE-CSE5 Testing [core]
CE-CSE6 Maintenance [core]
CE-CSE7 Project management [core]
CE-CSE8 Concurrent (hardware/software) design [core]
CE-CSE9 Implementation [elective]
CE-CSE10 Specialized systems [elective]
CE-CSE 11 Reliability and fault tolerance [elective]

CE-CSE0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying computer systems engineering
- Highlight some people that influenced or contributed to the area of computer systems engineering
- Explain briefly the concept of a system, a subsystem, the role of people, the recursive nature of this concept and relevance of this observation, the different disciplines involved, and the need for interdisciplinary approach
- Indicate how the life cycle contributes to systems engineering
- Mention that testing and maintenance are important systems engineering
- Indicate some important topic areas such as system-level design, hardware-software interface, direct and indirect interaction, and the human-computer interface
- Mention the nature of systems engineering including balancing costs, performance, and market considerations
- Indicate the importance of design decisions at the systems level and trade-offs
- Explain the need for flexibility and agility and reflection on approaches used and the competency of individuals
- Provide some examples of hardware-software trade-offs with illustrations and applications
- Indicate the importance and influence of the human computer interface in systems development
- Mention why it is important to know how to build reliable systems from unreliable components
- Indicate the importance and influence of standards, guidelines, legislation, regulations, and professional issues computer systems design
- Note that many computer systems designs become continually evolving systems
- Explore some additional resources associated with computer systems engineering
- Explain the purpose and role of computer systems engineering in computer engineering

Learning outcomes:
1. Identify some contributors to computer systems engineering and relate their achievements to the knowledge area.
2. Recognize and explain the possible interdisciplinary nature associated with the development of the range of computer-based systems.
3. Recognize and explain some of the mechanisms used in the development of computer based systems.
4. Recite some elements of the systems life cycle.
5. Describe how computer engineering uses or benefits from computer systems engineering.

CE-CSE1 Life cycle [core]
Minimum core coverage time: 2 hours

Topics:
- Nature of life cycle, role of life cycle model, quality in relation to the life cycle, influence of system size on choice of life cycle model and nature of system, agility issues
- Different models of the life cycle: Strengths and weaknesses of each
- The concept of process: Process improvement, basis for this is information, gathering information
- Maturity models, standards and guidelines

Learning outcomes:
1. Recognize the need for a disciplined approach to system development and explain the elements of this in particular contexts.
2. Explain how to gather data to inform process improvement.

CE-CSE2 Requirements analysis and elicitation [core]
Minimum core coverage time: 2 hours

Topics:
- System analysis: identification of need, feasibility considerations, economic considerations
- Nature of requirements : functional and non-functional requirements
- Approaches to determine requirements: analysis task, elements of this including communications and information gathering
- Prototyping, simulation and modeling
- Human factors
- Building expertise over time
- Role of experts and experience
- Non-functional requirements; the range of possibilities, the quantification issue
- Human factors issues: standards, user interface design
- Specific applications; building computer systems such as desktops, laptops, hand-held devices, digital cameras, mobile phones, video phones

Learning outcomes:
1. Describe the strengths and weaknesses of the major approaches to requirements elicitation and capture.
2. Apply one of a range of techniques to elicit and then describe the requirements for a particular system.

CE-CSE3 Specification [core]
Minimum core coverage time: 2 hours

Topics:
- Functional and non-functional specifications: different approaches and different possibilities
- Quality in relation to specification; completeness, consistency, simplicity, verifiability, basis for design; specification in the event of failure
- Test plans based on the specification: role of independence in relation to test; safety cases
- Limitations of such tests
- Degraded mode of operation: possibilities, testing in this context

Learning outcomes:
1. Recognize the characteristics of a high quality specification.
2. Assess the quality of a given specification.
3. Create a high quality specification of a given system.

CE-CSE4 Architectural design [core]
Minimum core coverage time: 3 hours

Topics:
- Basis for subdivision into systems and subsystems, basis for making these decisions
- Elements of high quality design
- Systems-level design strategies used in computer systems engineering including their strengths and weaknesses; inclusion of diagnostics in the event of failure; special problems of the hardware/software interface
- Design issues associated with achieving reliability; the role of redundancy; independence of designs, separation of concerns; specifications of subsystems, selection of subcontractor
- Different approaches to architectural design, their strengths and weaknesses
- Design to achieve performance measures such as in reliability and safety
- Concept of common cause failure
- Failure modes, approaches to fault tolerant design; dealing with failure

Learning outcomes:
1. Describe the strengths and weaknesses of the range of design decisions and methods.
2. Select and implement an appropriate approach to design for a range of possible applications.

CE-CSE5 Testing [core]
Minimum core coverage time: 2 hours

Topics:
- Nature of testing – do throughout life cycle – efficient and effective processes
- Test plans – purpose, nature
- White board, black board, regression testing, stress testing, interface testing
- Tool support to accommodate efficient and effective development including regression testing
- System-level test and diagnosis
- Printed circuit board, MCM, and core-based testing
- Software testing

Learning outcomes:
1. Recognize the range of tests appropriate for each stage of the systems life cycle.
2. Select an appropriate combination of tests for ensuring the quality of a system.

CE-CSE6 Maintenance [core]
Minimum core coverage time: 2 hours

Topics:
- Inevitability of maintenance in certain systems such as in hardware upgrade and tool development
- Patterns of behavior in relation to maintenance as in hardware, software, communications, and trends
- Measurement to inform maintenance as with bottlenecks
- Nature of maintenance: defect removal, upgrade, enhancement
- Impact analysis; decision making in relation to maintenance, configuration control board role
- Configuration management and version control in engineering systems – the need for this, the issues associated with it, the nature of the information to be held; legal requirements; planning for possible disasters
- Tool support and the nature of this
- Building expertise for later re-use; the issues, balances, options

Learning outcomes:
1. Understand the nature of maintenance in computer systems engineering.
2. Recognize and apply mechanisms that ensure design support maintenance.
3. Apply the principles in situations of modest complexity.

CE-CSE7 Project management [core]
Minimum core coverage time: 2 hours

Topics:
- The nature of project management in systems engineering, basic principles
- Composition of teams, difficulty of software project management
- Resource allocation
- Allocation of decision making to teams: Issues and options
- Gantt charts: Project planning, costing, teamwork
- Ensuring project management information; ensuring timely compliance with specification**;** timely delivery
- Standards, legal requirements, consultants subcontractors; their use and the management of these
- Role of metrics in support of management

Learning outcomes:
1. Recognize and know how to address the major problems of project management in computer engineering including multi disciplinary issues.
2. Describe the tools used to provide evidence to support all phases of the systems lifecycle.

CE-CSE8 Concurrent (hardware/software) design [core]
Minimum core coverage time: 2 hours

Topics:
- Applications areas with particular performance constraints that make the coordinated development of both hardware and software important as in speech coders and radio modems
- Demands of hard real-time features
- Hardware and software co-design

Learning outcomes:
1. Recognize the potential of hardware-software co-design in circumstances in which this approach is pertinent.
2. Apply hardware-software co-design principles in situations of modest complexity.

CE-CSE9 Implementation [elective]

Topics:

- Choosing technologies appropriate for particular purposes
- Rapid applications development
- Role of standards and documentation in relation to this
- Ensuring levels of performance, nature of tests, regression testing
- Technology specific issues

Learning outcomes:

1. Select technologies appropriate for achieving a high quality product over a range of applications.
2. Demonstrate the ability to implement at least one from a range of computer based systems

CE-CSE10 Specialized systems [elective]

Topics:

- Risk and hazard analysis; strategies for risk reduction, risk control – implications for implementation Role of preliminary hazard analysis
- Concept of integrity level: quantifying this and its impact on the life cycle issues
- Safety critical; concept of safety plan
- Security critical systems and other high integrity systems; high integrity functions: ensuring their performance
- Design based on these key functions required for achieving the required integrity level
- Range of strategies for achieving a variety of possible high performance levels; to include safety, reliability, security
- Choosing approaches throughout the life cycle appropriate to required integrity level
- International standards, legal requirements

Learning outcomes:

1. Recognize the special requirements of a range of specialist systems.
2. Demonstrate an ability to select approaches to the development of a range of specialist systems that are commensurate with the intended integrity level.

CE-CSE11 Reliability and fault tolerance [elective]

Topics:

- Reliability and availability modeling
- Hardware redundancy approaches
- Error detecting and correcting codes
- Software approaches to tolerating hardware faults
- Software reliability models
- Software fault-tolerance methods (N-version programming, recovery blocks, rollback and recovery)
- Fault tolerance in operating systems and data structures
- Fault tolerance in database and distributed systems
- Fault-tolerance in transaction processing systems
- Fault-tolerant systems for aerospace, telecommunications, and industrial control

Learning outcomes:

1. Understand the concepts of system reliability and availability, and their relationship to faults.
2. Understand basic redundancy approaches to fault tolerance to improve system reliability and/or availability.
3. Understand software faults, and redundancy methods used to tolerate them.
4. Understand fault tolerance, reliability, and availability requirements of different applications (database, aerospace, telecommunications, industrial control, transaction processing).

Circuits and Signals (CE-CSG)

CE-CSG0 History and overview [core]
CE-CSG1 Electrical quantities [core]
CE-CSG2 Resistive circuits and networks [core]
CE-CSG3 Reactive circuits and networks [core]
CE-CSG4 Frequency response [core]
CE-CSG5 Sinusoidal analysis [core]
CE-CSG6 Convolution [core]
CE-CSG7 Fourier analysis [elective]
CE-CSG8 Filters [elective]
CE-CSG9 Laplace transforms [elective]

CE-CSG0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying circuits and systems
- Highlight some people that influenced or contributed to the area of circuits and systems
- Indicate some important topic areas such as electrical quantities, resistance, reactance, frequency response, sinusoids, convolution, discrete-time signals, Fourier representation, filters, and transforms
- Contrast between current and voltage
- Describe Ohm's Law
- Explain reactive elements such as inductance and capacitance
- Indicate that frequency affects reactive elements, but not resistance elements
- Mention phase and the meaning of being "out of phase "
- Distinguish how signal sampling can produce aliasing and quantizing
- Illustrate the design of a "square wave" from a trigonometric Fourier series
- Explore some additional resources associated with circuits and systems
- Explain the purpose and role of circuits and systems in computer engineering

Learning outcomes:
1. Identify some contributors to circuits and systems and relate their achievements to the knowledge area.
2. Articulate the difference between resistance and reactance.
3. Articulate Ohm's Law.
4. Distinguish between inductance and capacitance.
5. Articulate the meaning of phase.
6. Describe aliasing.
7. Articulate the purpose of a Fourier series.
8. Describe how computer engineering uses or benefits from circuits and systems.

CE-CSG1 Electrical quantities [core]
Minimum core coverage time: 3 hours

Topics:
- Charge
- Current
- Voltage
- Energy
- Power

Learning outcomes:
1. Understand the concept of and to represent basic electrical quantities.
2. Understand the relationships between basic electrical quantities.

CE-CSG2 Resistive circuits and networks [core]
Minimum core coverage time: 9 hours

Topics:
- Ohm's Law
- Kirchoff's laws
- Independent and dependent sources
- Series and parallel elements
- Voltage and Current measurement
- Mesh and Nodal analysis
- Superposition
- Thevenin's and Norton's theorems
- Maximum power transfer

Learning outcomes:
1. Represent and manipulate basic resistive circuit equations.
2. Analyze and simplify basic resistive circuits.
3. Understand and use network analysis tools for resistive circuits.

CE-CSG3 Reactive circuits and networks [core]
Minimum core coverage time: 12 hours

Topics:
- Inductance
- Capacitance
- Mutual inductance
- Time constants for RL and RC circuits
- Transient response of RL, RC, and RLC circuits
- Damping
- Transformers

Learning outcomes:
1. Represent basic energy storage devices.
2. Understand how to combine various combinations of inductors and capacitors.
3. Understand simple transient response of various R, L, and C circuits.
4. Analyze and design simple R, L, and C circuits.

CE-CSG4 Frequency response [core]
Minimum core coverage time: 9 hours

Topics:
- Response of RL, RC, and RLC circuits
- Transfer functions
- Two-port circuits
- Parallel and series resonance

Learning outcomes:
1. Understand the frequency domain characteristics of electrical circuits.
2. Analyze and design frequency selective circuits.

CE-CSG5 Sinusoidal analysis [core]
Minimum core coverage time: 6 hours

Topics:
- Phasor representation of voltage and current
- Forced response to sinusoidal functions
- Impedance and Admittance
- Nodal and Mesh analysis
- Thevenin's and Norton's theorems
- Phasor diagrams
- Superposition
- Source transformations

Learning outcomes:
1. Understand response of electrical circuits to sinusoidal signal excitation.
2. Analyze circuits using the techniques given.

CE-CSG6 Convolution [core]
Minimum core coverage time: 3 hours

Topics:
- Impulse response
- Convolution integral
- Physically realizable systems
- Graphical methods

Learning outcomes:
1. Use the convolution technique to analyze circuits.
2. Represent convolution using graphical techniques.

CE-CSG7 Fourier analysis [elective]

Topics:
- Signal representation by Fourier series
- Trigonometric Fourier series
- Exponential Fourier series
- Definition of the Fourier transform
- Properties of the Fourier transform
- Circuit analysis using the Fourier transform

Learning outcomes:
1. Represent signals using Fourier series.
2. Understand the Fourier transform and its properties.
3. Apply Fourier transform techniques to circuit analysis.

CE-CSG8 Filters [elective]

Topics:
- Frequency selective circuits
- Transfer functions
- Passive filters
- Active filters

Learning outcomes:
1. Understand frequency selective circuits.
2. Design filters that have specified frequency characteristics.

CE-CSG9 Laplace transforms [elective]

Topics:
- Laplace transform integral
- Impulse response
- Step functions
- Ramp functions
- Inverse transforms
- Poles and zeroes
- Initial valve theorem
- Final valve theorem
- Circuit analysis using Laplace transforms

Learning outcomes:
1. Understand the Laplace transform technique and its mathematical representation.
2. Represent circuits and signals by their Laplace transforms.
3. Use the Laplace transform to describe electrical circuits and compute their behavior.

Database Systems (CE-DBS)

CE-DBS0 History and overview [core]
CE-DBS1 Database systems [core] *
CE-DBS2 Data modeling [core] *
CE-DBS3 Relational databases [elective] *
CE-DBS4 Database query languages [elective] *
CE-DBS5 Relational database design [elective] *
CE-DBS6 Transaction processing [elective] *
CE-DBS7 Distributed databases [elective] *
CE-DBS8 Physical database design [elective] *

* Consult the CC2001 Report [ACM/IEEECS, 2001] IM Knowledge Area for more detail

CE-DBS0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying database systems
- Highlight some people that influenced or contributed to the area of database systems
- Indicate some important topic areas such as information systems, database systems and, data modeling
- Contrast the meanings between data, information, and knowledge
- Describe a database system and its components
- Mention the use of database query languages
- Describe the meaning and purpose of a data model
- Explore some additional resources associated with database systems
- Explain the purpose and role of database systems in computer engineering

Learning outcomes:
1. Identify some contributors to database systems and relate their achievements to the knowledge area.
2. Explain how knowledge differs from information and data.
3. Name some components of a database system.
4. Name one query language.
5. Articulate the purpose of a data model.
6. Describe how computer engineering uses or benefits from database systems and information management.

CE-DBS1 Database systems [core]
Minimum core coverage time: 2 hours

Topics:
- Components of database systems; problem of the accuracy of information
- Database management system (DBMS) functions: the different possibilities and the role they play in a database system
- Database architectures: the possibilities, the concept of, the importance of and the reality of data independence
- Use of a database query language

Learning outcomes:
1. Explain the characteristics that distinguish the database approach from the traditional approach of programming with data files.
2. Cite the basic goals, functions, models, components, applications, and social impact of database systems.
3. Describe the components of a database system and give examples of their use.
4. Use a query language to elicit information from a database.

CE-DBS2 Data modeling [core]
Minimum core coverage time: 2 hours

Topics:
- Data modeling: the role of this, the benefits it brings and the common approaches; conceptual data model, physical data model, and representational data model
- Basic concepts: to include key, foreign key, record, relation
- Conceptual models: possibilities, entity-relationship model and UML; strengths and weaknesses; notational issues
- Object-oriented model: the main concepts and object identity, type constructors, encapsulation, inheritance, polymorphism, and versioning; basic approaches
- Relational data model: basic terminology, basic approaches, strengths and weaknesses

Learning outcomes:
1. Categorize data models based on the types of concepts that they provide to describe the database structure.
2. Compare and contrast the basic principles of the relational data model and those of the object-oriented model as they apply to computer engineering applications.

CE-DBS3 Relational databases [elective]

Topics:
- Concepts of conceptual schema and relational schema: uses, comparisons, mapping conceptual schema to relational schema
- Entity integrity constraint and referential integrity constraint; definitions, uses
- Relational algebra and relational calculus
- Relational algebra operations from mathematical set theory (*union, intersection, difference,* and *Cartesian product*) contrasted with the relational algebra operations developed specifically for relational databases (*select, product, join,* and *division*).

Learning outcomes:
1. Prepare a relational schema from a conceptual model developed using the entity-relationship model
2. Explain and demonstrate the concepts of entity integrity constraint and referential integrity constraint (including the definition of the concept of a foreign key).
3. Demonstrate the successful formulation of queries in the relational algebra and queries in the tuple relational calculus.

CE-DBS4 Database query languages [elective]

Topics:
- Overview of database languages
- Structured query language (SQL): fundamental concepts including data definition, query formulation, update sub-language, constraints, integrity
- Query processing strategies; query optimization
- Query by example and 4th-generation environments
- Embedding non-procedural queries in a procedural language
- Introduction to object query language

Learning outcomes:
1. Create a relational database schema in SQL that incorporates key, entity integrity, and referential integrity constraints.
2. Demonstrate data definition in SQL and retrieving information from a database using the SQL *SELECT* statement
3. Evaluate a set of query processing strategies and select the optimal strategy
4. Demonstrate the ability to embed object-oriented queries into an appropriate stand-alone programming language

CE-DBS5. Relational database design [elective]

Topics:
- Database design
- The concept of functional dependency
- Normal forms: first, second, third and Boyce-Codd normal forms; motivation for each of these, applicability; mechanisms for producing these normal forms
- Multi-valued dependency: fourth normal form; join dependency; fifth normal form
- Representation theory

Learning outcomes:
1. Determine the functional dependency between two or more attributes that are a subset of a relation.
2. Demonstrate the ability to transform a relation into a prescribed normal form
3. Explain the impact of normalization on the efficiency of database operations and query optimization.
4. Describe a multi-valued dependency and the type of constraints it specifies.

CE-DBS6 Transaction processing [elective]

Topics:

- Transactions: the purpose and the nature of transactions; creating a transaction using SQL; characteristics of efficient transaction execution; concept of commit
- Failure and recovery: the differing possibilities, their strengths and weaknesses
- Concurrency control: the special problems introduced by concurrency; the solution to these problems; isolation levels and their effects

Learning outcomes:

1. Explain the purpose of rollback and the way logging assures that proper rollback takes place.
2. Outline the special problems arising from the use of concurrency and the effect of effect of different isolation levels on the concurrency control mechanisms.
3. Demonstrate the ability to apply suitable ideas on failure and recovery to an application taken from computer engineering.

CE-DBS7 Distributed databases [elective]

Topics:

- Advantages offered by the introduction of distribution: the problems introduced
- Distributed data storage: techniques used for data fragmentation, replication, and allocation during the distributed database design process
- Distributed query processing: strategies for executing distributed queries
- Distributed transaction model
- Concurrency control: the different approaches including those based on the distinguished copy techniques and the voting method
- Homogeneous and heterogeneous solutions
- The client-server approach

Learning outcomes:

1. Evaluate simple strategies for executing a distributed query to select the strategy that minimizes the amount of data transfer.
2. Explain how to use the two-phase commit protocol to deal with committing a transaction that accesses databases stored on multiple nodes.
3. Compare and contrast the different approaches to distributed concurrency control.

CE-DBS8 Physical database design [elective]

Topics:

- Storage requirements for a range of data including characters, numbers, strings, text, sound, video and file structure
- Characteristics of storage to support a range of databases including use of CDs, memory in machines of different kinds; nature of the storage systems involved and the factors influencing choice
- Records and record types; fixed length and variable length; storage organization
- Files of different kinds and file structures: sequential files, indexed files, hashed files; signature files; files with dense index
- B-trees; definition, use in particular in implementing dynamic multi-level indices
- Data compression: reasons for using this, compression algorithms, strengths and weaknesses of each approach; software support
- Database efficiency and tuning: performance measures, monitoring performance

Learning outcomes:

1. Give examples of the application of primary, secondary, and clustering indexes.
2. Explain the theory and application of internal and external hashing techniques and its use in facilitating dynamic file expansion.
3. Describe the relationships among hashing, compression, and efficient database searches.
4. Explain how physical database design affects database transaction efficiency.

Digital Logic (CE-DIG)

CE-DIG0 History and overview [core]
CE-DIG1 Switching theory [core]
CE-DIG2 Combinational logic circuits [core]
CE-DIG3 Modular design of combinational circuits [core]
CE-DIG4 Memory elements [core]
CE-DIG5 Sequential logic circuits [core]
CE-DIG6 Digital systems design [core]
CE-DIG7 Modeling and simulation [core]
CE-DIG8 Formal verification [core]
CE-DIG9 Fault models and testing [core]
CE-DIG10 Design for testability [elective]

CE-DIG0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying digital logic
- Highlight some people that influenced or contributed to the area of digital logic
- Indicate some important topic areas such as logic circuits, switching, memory, registers, and digital systems
- Highlight the importance of Boolean logic to the knowledge area
- Mention the meaning and importance of sequential logic
- Contrast the meanings of gates, circuits, combinational circuits, and modules
- Indicate that memory is a logical circuit
- Highlight that a special form of memory module forms registers
- Mention how systems result from modules and circuits
- Explore some additional resources associated with digital logic
- Explain the purpose and role of digital logic in computer engineering

Learning outcomes:
1. Identify some contributors to digital logic and relate their achievements to the knowledge area.
2. Explain why Boolean logic is important to this subject.
3. Articulate why gates are the fundamental elements of a digital system.
4. Contrast the difference between a memory element and a register.
5. Indicate some uses for sequential logic.
6. Describe how computer engineering uses or benefits from digital logic.

CE-DIG1 Switching theory [core]
Minimum core coverage time: 6 hours

Topics:
- Number systems and codes
- Binary arithmetic
- Boolean and switching algebra
- Representation and manipulation of switching functions
- Minimization of switching functions
- Incompletely specified switching functions

Learning outcomes:
1. Work with binary number systems and arithmetic.
2. Derive and manipulate switching functions that form the basis of digital circuits.
3. Reduce switching functions to simplify circuits used to realize them.

CE-DIG2 Combinational logic circuits [core]
Minimum core coverage time: 4 hours

Topics:
- Basic logic gates (AND,OR,NOT,NAND,NOR,XOR)
- Realization of switching functions with networks of logic gates
- 2-level networks: AND-OR,OR-AND,NAND-NAND,NOR-NOR
- Multi-level networks
- Physical properties of logic gates (technology, fan-in, fan-out, propagation delay)
- Elimination of timing hazards/glitches

Learning outcomes:
1. Realize switching functions with networks of logic gates.
2. Explain and apply fundamental characteristics of relevant electronic technologies, such as propagation delay, fan-in, fan-out, and power dissipation and noise margin.

CE-DIG3 Modular design of combinational circuits [core]
Minimum core coverage time: 6 hours

Topics:
- Design of medium scale combinational logic modules
- Multiplexers, demultiplexers, decoders, encoders, comparators
- Arithmetic functions (adders, subtracters, carry lookahead)
- Multipliers, dividers
- Arithmetic and logic units (ALUs)
- Hierarchical design of combinational circuits using logic modules

Learning outcomes:
1. Analyze and explain uses of small- and medium-scale logic functions as building blocks.
2. Analyze and design combinational logic networks in a hierarchical, modular approach, using standard and custom logic functions.

CE-DIG4 Memory elements [core]
Minimum core coverage time: 3 hours

Topics:
- Unclocked and clocked memory devices (latches, flip flops)
- Level vs. edge-sensitive, and master-slave devices
- Basic flip flops (SR, D, JK, T)
- Asynchronous flip flop inputs (preset, clear)
- Timing constraints (setup time, hold time) and propagation delays
- Data registers (selection, clocking, timing)
- Random-access memory (RAM)

Learning outcomes:
1. Design and describe the operation of basic memory elements.
2. Analyze circuits containing basic memory elements.
3. Apply the concepts of basic timing issues, including clocking, timing constraints, and propagation delays during the design process.

CE-DIG5 Sequential logic circuits [core]
Minimum core coverage time: 10 hours

Topics:
- Finite state machines (FSMs), clocked and unclocked
- Mealy vs. Moore models of FSMs
- Modeling FSM behavior: State diagrams and state tables, timing diagrams, algorithmic state machine charts
- Analysis of synchronous and asynchronous circuits
- Design of synchronous sequential circuits: State minimization, state assignment, next state and output equation realization
- Sequential functional units: Data registers, shift registers, counters, sequence detectors, synchronizers, debouncers, controllers

Learning outcomes:
1. Analyze the behavior of synchronous and asynchronous machines.
2. Synthesize synchronous and asynchronous sequential machines.

CE-DIG6 Digital systems design [core]
Minimum core coverage time: 12 hours

Topics:
- Hierarchical, modular design of digital systems
- Synthesis of digital circuits from HDL models
- Design principles and techniques: Bridging conceptual levels – top down/bottom up, divide and conquer, iteration, satisfying a behavior with a digital structure
- Functional units, building blocks and LSI components: Adder, shifter, register, ALU, and control circuits, tri-state devices and buses
- Control concepts: Register transfer notation, major control state, sequences of micro-operations, conditional execution of micro-operations
- Timing concepts: System timing dependencies, sequencing, clock generation, distribution, and skew
- Programmable logic devices (PLDs) and field-programmable gate arrays (FPGAs), PLAs, ROMs, PALs, complex PLDs

Learning outcomes:
1. Apply digital system design principles and descriptive techniques.
2. Analyze and design functional building blocks and control and timing concepts of digital systems.
3. Develop a complex digital system design in a hierarchical fashion using top-down and bottom-up design approaches.
4. Utilize programmable devices such as FPGAs and PLDs to implement digital system designs.

CE-DIG7 Modeling and simulation [core]
Minimum core coverage time: 5 hours

Topics:
- Schematic capture
- Hierarchical schematic modeling for complex systems
- Digital system modeling with hardware description languages (VHDL, Verilog)
- Other modeling techniques (timing diagrams, register transfer languages, state diagrams, algorithmic state machines)
- Functional simulation of combinational and sequential circuits
- Timing models of digital circuit elements: Propagation delay, rise/fall time, setup and hold times, pulse widths
- Timing simulation to measure delays and study signals subject to timing constraints
- Simulation test-bench design

Learning outcomes:
1. Model and simulate a digital system using schematic diagrams.
2. Model and simulate a digital system using a hardware description language, such as VHDL or Verilog.
3. Understand timing issues in digital systems and know how to study these via digital circuit simulation.

CE-DIG8 Formal verification [core]
Minimum core coverage time: 5 hours

Topics:
- Relationship of good design practice to formal verification
- Comparison and contrast of formal verification, validation, testing, and reliability
- Verification by model checking
- Verification by proofs
- Verification by equivalence checking
- Verification by simulation and test-benches
- Verification by assertions and verification languages
- Verification by testing
- Economics of verification
- Other verification: signal integrity, specification, reliability, safety, power, cooling

Learning outcomes:
1. Understand the difference between good design practice and formal verification.
2. Distinguish between the various forms of verification.

CE-DIG9 Faults models and testing [core]
Minimum core coverage time: 5 hours

Topics:
- Logical (stuck-at) faults (single and multiple)
- Other fault models (bridging, opens, delay faults)
- Yield and defect levels
- Test coverage
- Fault equivalence and dominance
- Fault simulation and fault grading
- Test generation algorithms such as the D-algorithm and PODEM
- Automatic Test Pattern Generation (ATPG): Pseudorandom techniques, deterministic test pattern generation
- Test generation algorithms for sequential circuits
- Memory testing
- PLA testing

Learning outcomes:
1. Understand the types and characteristics of the most common faults that occur in digital circuits.
2. Understand the concept of test coverage, and be able to design a test to achieve high test coverage for stuck-at faults.
3. Understand the role of computer-aided testing tools, including fault simulation and ATPG.
4. Understand basic approaches to testing memory devices and PLAs.

CE-DIG10 Design for testability [elective]

Topics:
- Testability measures (controllability, observability)
- Scan and partial scan design
- BIST and other design for testability techniques
- Boundary scan and the IEEE 1149.1 testability standard
- Ad-hoc methods

Learning outcomes:
1. Understand measures of testability to appreciate what to do to improve testability.
2. Understand scan design and some of the other basic methods used to improve testability in digital circuits.
3. Understand the concept of built-in self test and some of the basic BIST approaches used in digital circuits.

Discrete Structures (CE-DSC)

CE-DSC0 History and overview
CE-DSC1 Functions, relations, and sets [core] *
CE-DSC2 Basic logic [core] *
CE-DSC3 Proof techniques [core] *
CE-DSC4 Basics of counting [core] *
CE-DSC5 Graphs and trees [core] *
CE-DSC6 Discrete probability [core] *
CE-DSC7 Recursion [elective] *

* Consult the CC2001 Report [ACM/IEEECS, 2001] DS Knowledge Area for more detail

CE-DSC0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Knowledge themes include sets, logic, functions, and graphs
- Contributors to the subject
- Purpose and role of discrete structures in computer engineering
- Contrasts between discrete-time models vs. continuous-time models

Learning outcomes:
1. Associate the themes involved with discrete structures.
2. Identify contributors to the subject area.
3. Articulate differences between discrete and continuous models.
4. Describe how computer engineering could make use of discrete structures.

CE-DSC1 Functions, relations, and sets [core]
Minimum core coverage time: 6 hours

Topics:
- Functions (one-to-one, onto, inverses, composition)
- Relations (reflexivity, symmetry, transitivity, equivalence relations)
- Discrete versus continuous functions and relations
- Sets (Venn diagrams, complements, Cartesian products, power sets)
- Cardinality and countability

Learning outcomes:
1. Illustrate by examples the basic terminology of functions, relations, and sets.
2. Illustrate by examples, both discrete and continuous, the operations associated with sets, functions, and relations.
3. Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context.
4. Apply functions and relations to problems in a computer engineering setting.

CE-DSC2 Basic logic [core]
Minimum core coverage time: 10 hours

Topics:
- Propositional logic
- Logical connectives
- Truth tables
- Use of logic to illustrate connectives
- Normal forms (conjunctive and disjunctive)
- Predicate logic
- Universal and existential quantification
- Limitations of predicate logic
- Boolean algebra
- Applications of logic to computer engineering

Learning outcomes:
1. Manipulate formal methods of symbolic propositional and predicate logic.
2. Use formal tools of symbolic logic.
3. Demonstrate knowledge of formal logic proofs and logical reasoning through solving problems such as puzzles.
4. Apply logic gates to problems in logic.

CE-DSC3 Proof techniques [core]
Minimum core coverage time: 6 hours

Topics:
- Notions of implication, converse, inverse, negation, and contradiction
- The structure of formal proofs
- Direct proofs
- Proof by counterexample, contraposition, and contradiction
- Mathematical induction and strong induction

Learning outcomes:
1. Outline basic proofs for theorems using the techniques of proof by contradiction and mathematical induction.
2. Solve problems by different methods of proof.
3. Apply proof techniques to problems in a computer engineering setting.

CE-DSC4 Basics of counting [core]
Minimum core coverage time: 4 hours

Topics:
- Permutations and combinations
- Counting arguments rule of products, rule of sums
- The pigeonhole principle
- Generating functions
- Applications to computer engineering

Learning outcomes:
1. Compute permutations and combinations of a set.
2. Interpret the meaning in the context of a particular counting application.
3. Show the application of the pigeonhole principle.
4. Apply counting principles to problems in a computer engineering setting.

CE-DSC5 Graphs and trees [core]
Minimum core coverage time: 4 hours

Topics:
- Trees
- Undirected graphs
- Directed graphs
- Spanning trees
- Shortest path
- Euler and Hamiltonian cycles
- Traversal strategies

Learning outcomes:
1. Illustrate by example the basic terminology of graph theory.
2. Show some of the properties and special cases of graph principles.
3. Construct models in computing using graphs and trees.
4. Relate graphs and trees to data structures, algorithms, and counting.
5. Apply graphs and trees to problems in a computer engineering setting.

CE-DSC6 Recursion [core]
Minimum core coverage time: 2 hours

Topics:
- Recursive mathematical definitions
- Developing recursive equations
- Solving recursive equations
- Applications of recursion to computer engineering

Learning outcomes:
1. Solve a variety of basic recursive equations.
2. Analyze a problem to create relevant recurrence equations where applicable.
3. Use a problem to identify important counting questions.
4. Relate the ideas of mathematical induction to recursion and recursively defined structures.
5. Apply recursion to problems in a computer engineering setting.

Digital Signal Processing (CE-DSP)

CE-DSP0 History and overview [core]
CE-DSP1 Theories and concepts [core]
CE-DSP2 Digital spectra analysis [core]
CE-DSP3 Discrete Fourier transform [core]
CE-DSP4 Sampling [core]
CE-DSP5 Transforms [core]
CE-DSP6 Digital filters [core]
CE-DSP7 Discrete time signals [elective]
CE-DSP8 Window functions [elective]
CE-DSP9 Convolution [elective]
CE-DSP10 Audio processing [elective]
CE-DSP11 Image processing [elective]

CE-DSP0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying digital signal processing and multimedia
- Highlight some people that influenced or contributed to the area of digital signal processing and multimedia
- Indicate some important topic areas such as digital audio, multimedia, wave tables, digital filters, image display, chromatic and achromatic lighting, and thresholds
- Contrast the meanings of analog and digital signals
- Explain the need for using transforms and why they are different for analog and discrete situations
- Indicate how the subject relates to simple graphics
- Contrast image processing from computer graphics
- Mention some techniques used in transformations such as Fourier, Laplace, and wavelet transforms
- Explore some additional resources associated with digital signal processing and multimedia
- Explain the purpose and role of digital signal processing and multimedia in computer engineering

Learning outcomes:
1. Identify some contributors to digital signal processing and multimedia and relate their achievements to the knowledge area.
2. Know the difference between analog and discrete signals.
3. Articulate the difference between image processing and computer graphics.
4. Indicate some of the characteristics of filters, in particular low- and high-pass filters.
5. Describe how computer engineering uses or benefits from digital signal processing and multimedia.

CE-DSP1 Theories and concepts [core]
Minimum core coverage time: 1 hour

Topics:
- The sampling theorem
- Nyquist frequency
- Aliasing
- Relationship between time and frequency domain
- Principle of causality such as discrete and continuous spectra

Learning outcomes:
1. Describe the sampling theorem
2. Distinguish between a time domain and a frequency domain
3. Contrast examples of discrete spectra and continuous spectra problems

CE-DSP2 Digital spectra analysis [core]
Minimum core coverage time: 1 hour

Topics:
- Spectral views
- Spectrum analysis
- Spectra of periodic signals
- Spectra of the impulse and a square wave
- Filtering
- Interpolation

Learning outcomes:
1. Describe the spectra of a periodic signal.
2. Contrast between the spectra of an impulse and a square wave.
3. Describe the importance of filtering.

CE-DSP3 Discrete Fourier transform [core]
Minimum core coverage time: 1 hour

Topics:
- Definition of the Discrete Fourier Transform (DFT)
- Relationship between original and transformed domains
- Algorithms of the DFT
- Linear convolutions
- Contrast DFT with the Fourier Transform and the Fast Fourier Transform (FFT)
- Filtering using DFT
- Filtering of long data sequences

Learning outcomes:
1. Explain the purpose of a Fourier transform in signal processing.
2. Describe the advantage of the FFT.
3. Explain the difference between the FFT and the DFT.
4. Understand how the DFT accomplishes filtering.

CE-DSP4 Sampling [core]
Minimum core coverage time: 1 hour

Topics:
- Implications of assumptions of repeated time series
- Group sampling of time signals
- Size of group and how it affects spectra
- Sampled signals
- Periodic signals
- Non-periodic signals
- Spectrograms

Learning outcomes:
1. Describe the advantages and disadvantages of using increased sampling rates.
2. Describe advantages of group sampling of time signals.
3. Contrast how group size affects signal spectra.
4. Indicate some of the advantages of sampling periodic signals.
5. Indicate some of the challenges of sampling non-periodic signals.

CE-DSP5 Transforms [core]
Minimum core coverage time: 4 hours

Topics:
- Concept and properties of the z–transform
- Inverse z–transforms
- Difference equations
- The Discrete Fourier Transform
- The Inverse DFT
- The Fast Fourier Transform Class
- The Inverse FFT method
- Fast Convolution using the FFT
- Power Spectral Density
- Frequency shifting using the FFT
- Filtering using FFT
- Additive and subtractive synthesis

Learning outcomes:
1. Understand the concept, properties and uses of the z–transform.
2. Understand the relationship between z–transform and the conformal map
3. Understand the Discrete Fourier transform and its significance.
4. Understand the Fast Fourier transform and its significance.
5. Understand the difference between additive and subtractive synthesis.
6. Understand the role of the FFT in additive and subtractive synthesis.

CE-DSP6 Digital filters [core]
Minimum core coverage time: 1 hour

Topics:
- Frequency response of discrete – time systems
- Recursive filter design
- Nonrecursive filter design
- Windowing
- FIR filters, frequency and phase response, time domain multi-tap filters, surface acoustic wave filters
- Poles and zeros in the z plane
- IIR filters, frequency and phase response
- Design of IIR Filters

Learning outcomes:
1. Understand frequency selective filters in the z–transform domain.
2. Design digital filters that have specified frequency characteristics.

CE-DSP7 Discrete time signals [elective]
Minimum core coverage time: 4 hours

Topics:
- Representation of signals
- Sampling of signals
- Quantizing
- Aliasing
- Difference Equations

Learning outcomes:
1. Understand the discrete-time representation of signals.
2. Understand errors introduced by sampling and quantizing.

CE-DSP8 Window functions [elective]

Topics :
- Definition of a window function
- Purpose of a window function
- Signal compression and transform properties
- Window functions and their impact on the spectra
- Window functions and the DFT

Learning outcomes:
1. Understand the definition of a window function.
2. Explain why window functions are important to digital signal processing.
3. Explain how window functions improve transform properties.

CE-DSP9 Convolution [elective]

Topics:
- Impulse response
- Convolution integral
- Physically realizable systems
- Graphical methods

Learning outcomes:
1. Use the convolution technique to analyze circuits.
2. Represent convolution using graphical techniques.

CE-DSP10 Audio processing [elective]

Topics:
- Speech coding
- Audio coding and MPEG algorithms
- Speech and audio enhancements
- Adaptive noise cancellation
- Speech recognition

Learning outcomes:
1. Describe the purpose of speech coding
2. Describe how digital techniques enhance speech and audio signals.
3. Explain how digital techniques cancel noise in audio processing.

CE-DSP11 Image processing [elective]

Topics:
- Analog to digital transformations
- Sampling and image integrity
- Smoothing images and low-pass filters
- Reconstruction and enhancement filtering
- Noise and images
- Spatial frequency

Learning outcomes:
1. Describe how sampling affects image integrity.
2. Explain how low-pass filtering tends to smooth out images.
3. Contrast between reconstruction and enhancement filters.
4. Describe way in which one can minimize image noise.

Electronics (CE-ELE)

CE-ELE0 History and overview [core]
CE-ELE1 Electronic properties of materials [core]
CE-ELE2 Diodes and diode circuits [core]
CE-ELE3 MOS transistors and biasing [core]
CE-ELE4 MOS logic families [core]
CE-ELE5 Bipolar transistors and logic families [core]
CE-ELE6 Design parameters and issues [core]
CE-ELE7 Storage elements [core]
CE-ELE8 Interfacing logic families and standard buses [core]
CE-ELE9 Operational amplifiers [core]
CE-ELE10 Circuit modeling and simulation [core]
CE-ELE11 Data conversion circuits [elective]
CE-ELE12 Electronic voltage and current sources [elective]
CE-ELE13 Amplifier design [elective]
CE-ELE14 Integrated circuit building blocks [elective]

CE-ELE0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying electronics
- Highlight some people that influenced or contributed to the area of electronics
- Indicate some important topic areas such as material properties, diodes and transistors, storage elements, interfaces and buses, operational amplifiers, and circuit simulators
- Contrast the meanings transistors and diodes
- Mention some issues and parameters in electronics design
- Describe the difference between an ordinary amplifier and an operational amplifier
- Mention the importance of data conversion and the circuits for doing the same
- Indicate some circuit simulators and contrast the advantages of each
- Explore some additional resources associated with electronics
- Explain the purpose and role of electronics in computer engineering

Learning outcomes:
1. Identify some contributors to electronics and relate their achievements to the knowledge area.
2. Describe a transistor and its functionality.
3. Identify some storage elements.
4. Articulate the purpose of buses.
5. Indicate the importance of designing data conversion circuits.
6. Identify two software products used for designing and simulating circuits.
7. Describe how computer engineering uses or benefits from electronics.

CE-ELE1 Electronic properties of materials [core]
Minimum core coverage time: 3 hours

Topics:
- Solid-state materials
- Electrons and holes
- Doping, acceptors and donors
- p- and n-type material
- Conductivity and resistivity
- Drift and diffusion currents, mobility and diffusivity

Learning outcomes:
1. Indicate the properties of materials that lead to be useful for the construction of electronic circuits, giving reasons.
2. Explain the uses of one particular material (as opposed to alternatives) to serve a stated purpose.

CE-ELE2 Diodes and diode circuits [core]
Minimum core coverage time: 5 hours

Topics:
- Diode operation and i-v characteristics
- Regions of operation, models, and limitations
- Schottky, Zener, variable capacitance diodes
- Single diode circuits, the load line
- Multi-diode circuits
- Rectifiers
- dc/dc converters
- Diode logic: AND and OR functions

Learning outcomes:
1. Explain the properties of diodes.
2. Outline the use of diodes in the construction of a range of circuits including rectifiers, ac/dc converters, and common logic functions.

CE-ELE3 MOS transistors and biasing [core]
Minimum core coverage time: 3 hours

Topics:
- NMOS field-effect transistor operation
- i-v characteristics
- Regions of operation, models, and limitations
- Enhancement and depletion-mode devices
- PMOS devices
- Transfer characteristic of FET with load resistor
- Biasing for logic and amplifier applications

Learning outcomes
1. Indicate the areas of use of NMOS, PMOS, CMOS, and dynamic logic families.
2. Demonstrate the ability to implement a range of logic functions using each of NMOS, PMOS, CMOS, and dynamic logic.

CE-ELE4 MOS logic families [core]
Minimum core coverage time: 7 hours

Topics:
- Logic level definitions
- NMOS logic design: Inverter, NOR, NAND, SOP, POS, complex gates
- PMOS logic
- CMOS logic: Inverter, NOR, NAND, SOP, POS, complex gates
- Dynamic logic
- CVS logic
- Cascade buffers
- NMOS and CMOS power/delay scaling

Learning outcomes
1. Explain the differences between the different MOS logic families.
2. Articulate the advantages of dynamic logic.

CE-ELE5 Bipolar transistors and logic families [core]
Minimum core coverage time: 4 hours

Topics:
- npn and pnp transistor operation
- i-v characteristics
- Regions of operation, models, and limitation
- Transfer characteristic of BJT with load resistor
- Biasing for logic and amplifier applications
- Logic level definitions
- The differential pair as a current switch
- Transistor-transistor logic – inverters, NAND, other functions
- Emitter-coupled logic – OR/NOR gate, other functions
- Low voltage bipolar logic families

Learning outcomes:
1. Indicate the areas of use of bipolar logic families.
2. Demonstrate the ability to implement a range of logic functions using bipolar logic.

CE-ELE6 Design parameters and issues [core]
Minimum core coverage time: 4 hours

Topics:
- Switching energy, power-delay product comparison,
- Propagation delay, rise time, fall time
- Fan-in and fan-out
- Power dissipation, noise margin
- Power supply distribution
- Sources of signal coupling and degradation
- Transmission line effects; passive, active, dc and ac termination
- Element tolerances
- Worst-case analysis of circuits
- Monte Carlo analysis
- Monte Carlo analysis in SPICE
- Six-sigma design

Learning outcomes:
1. Incorporate design strategies in power distributions and transmission.
2. Apply methods to minimize noise and other signal degradations.

CE-ELE7 Storage elements [core]
Minimum core coverage time: 3 hours

Topics:
- Latches
- Flip-flops
- Static RAM cells
- Dynamic RAM cells
- Sense amplifiers

Learning outcomes:
1. Compare and contrast the properties of different kinds of storage element to serve different purposes.
2. Select (with reasons) appropriate kinds of storage elements for use in a range of possible devices.

CE-ELE8 Interfacing logic families and standard buses [core]
Minimum core coverage time: 3 hours

Topics:
- Terminal characteristics of various logic families
- Standard interface characteristics
- Level translations: TTL/CMOS, TTL/ECL, CMOS/ECL
- Single-ended to differential and differential to single-ended conversion
- Transmission line characteristics, reflections
- Bus termination: Passive, active, dc, ac
- 4-20 mA current interfaces
- RS-XXX buses
- IEEE-XXXX buses
- Low-level differential signaling
- RAMBUS
- DDR

Learning outcomes:
1. Explain the practical difficulties resulting from the distribution of signals.
2. Explain ways to overcome these difficulties when interfacing different logic families.

CE-ELE9 Operational amplifiers [core]
Minimum core coverage time: 4 hours

Topics:
- Ideal op-amps and circuit analysis
- Ideal op-amp circuits: Inverting and non-inverting amplifiers, summing and difference amplifiers, integrator, low pass filter
- Non-ideal op-amps: dc errors, CMRR, input and output resistances, frequency response, output voltage and current limitations
- Circuits with non-ideal amplifiers
- Multi-stage op-amp circuits

Learning outcomes:
1. Explain with justification the ideal properties of operational amplifiers.
2. Design various amplifier structures and filters with ideal op-amps.
3. Understand characteristics of non-ideal op-amps.
4. Design simple circuits with them.

CE-ELE10 Circuit modeling and simulation [core]
Minimum core coverage time: 3 hours

Topics:
- DC analysis
- AC analysis
- Transient analysis
- Simulation control options
- Built-in solid-state device models
- Device parameter control
- Libraries
- Mixed-mode simulation

Learning outcomes:
1. Explain with justification the benefits and the drawbacks associated with the simulation of circuits.
2. Identify aspects of circuits that are not readily amenable to simulation.
3. Simulate a range of possible circuits using a suitable software package.

CE-ELE11 Data conversion circuits [elective]

Topics:
- D/A Converters: Definitions such as for codes, LSB, and MSB; linearity, differential linearity, offset and gain errors; weighted resistor D/A converter; R/2R ladders and D/A converters; weighted current source converters; delta-sigma converters
- A/D Converters: Definitions such as for codes, LSB, MSB, and missing codes; linearity, differential linearity, offset and gain errors, missing codes; counting converter; successive approximation; single and dual slope converters; flash converters; delta-sigma converters
- Sample-and-hold circuits

Learning outcomes:

1. Describe the properties that distinguish particular kinds of converters.
2. Given a range of possible scenarios, select (with justification) a converter appropriate for the scenario.

CE-ELE12 Electronic voltage and current sources [elective]

Topics:

- Electronic voltage sources: ideal voltage source characteristics; voltage references; emitter followers; voltage sources utilizing operational amplifiers
- Electronic current sources: ideal current source characteristics; transistor current sources; common-emitter, cascode, regulated cascode circuits; current sources utilizing operational amplifiers

Learning outcomes:

1. Explain the various types of electronic voltage sources.
2. Explain the various types of electronic current sources.

CE-ELE13 Amplifier design [elective]

Topics:

- Characteristics and properties of a linear amplifier: voltage gain, current gain, power gain, dB scale, frequency domain characteristics, distortion
- Definition of small-signal in diodes and transistors
- Bias circuits for linear amplification, voltage, current, power gain, input/output resistances
- Amplifier configurations: BJT common-emitter, common-base and common-collector; MOSFET common-source, common-gate, common-drain
- Low frequency response, high frequency device models, high frequency response; short-circuit and open-circuit time constant techniques
- Multistage transistor amplifiers: ac and dc coupled amplifiers; frequency response
- Differential pairs such as MOSFET and BJT
- Current sources and biasing; current mirrors; active loads
- Elementary two- and three-stage op-amp circuits
- Classical op-amp input stages

Learning outcomes:

1. Explain the properties of linear amplifiers (as opposed to other kinds of amplifiers) and to identify significant uses.
2. Demonstrate an ability to design and build a range of possible linear amplifiers.

CE-ELE14 Integrated circuit building blocks [elective]

Topics:

- Power circuits: class A output stages; class B and class B push-pull output stages; cross over distortion; class AB amplifiers; power semiconductor devices; switching (boost and buck) converters
- Active filters: their properties and characteristics
- Continuous time filter: bandwidth, Q; single op-amp active filters; multi op-amp filters; Q and cutoff/center frequency sensitivity
- Switched capacitor filters
- Oscillators: Barkhausen criteria for oscillation; RC oscillators; LC oscillators Colpitts, Hartley; crystal oscillators; multivibrators
- Operational amplifiers and circuits; comparators; PTAT circuits; band-gap references; voltage regulators; Gilbert multipliers
- Circuits for wireless applications: noise; noise; passive components; low noise amplifiers; frequency conversion and mixers; power amplifiers – Class B, Class C

Learning outcomes:

1. Explain the properties and the nature of the common building blocks used to build integrated circuits.
2. Design and assemble integrated circuit building blocks to provide circuits for a range of applications including wireless applications.

Embedded Systems (CE-ESY)

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

CE-ESY0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying embedded systems
- Highlight some people that influenced or contributed to the area of embedded systems
- Indicate some important topic areas such as mapping between language and hardware, classifications, influence of software engineering, applications and techniques, and tool support
- Contrast between an embedded system and other computer systems
- Mention the role of programming and its associated languages as applied to embedded systems
- Explore some additional resources associated with embedded systems
- Explain the purpose and role of embedded systems in computer engineering

Learning outcomes:
1. Identify some contributors to embedded systems and relate their achievements to the knowledge area.
2. Describe the meaning of an embedded system.
3. Explain the reasons for the importance of embedded systems.
4. Describe the relationship between programming languages and embedded systems..
5. Describe how computer engineering uses or benefits from embedded systems.

CE-ESY1 Embedded microcontrollers [core]
Minimum core coverage time: 6 hours

Topics:
- Structure of a basic computer system: CPU, memory, I/O devices on a bus
- CPU families used in microcontrollers: 4-bit, 8-bit, 16-32-bit
- Basic I/O devices: timers/counters, GPIO, A/D, D/A
- Polled I/O vs. interrupt-driven I/O
- Interrupt structures: vectored and prioritized interrupts
- DMA transfers
- Memory management units
- Memory hierarchies and caches

Learning outcomes:
1. Understand the CPU in the context of a complete system with I/O and memory.
2. Understand how the CPU talks to the outside world through devices.
3. Understand how memory system design (caches, memory management) affect program design and performance.

CE-ESY2 Embedded programs [core]
Minimum core coverage time: 3 hours

Topics:
- The program translation process: compilation, assembly, linking
- Representations of programs: data flow and control flow
- Fundamental concepts of assembly language and linking: labels, address management
- Compilation tasks: mapping variables to memory, managing data structures, translating control structures, and translating expressions
- What can and cannot be controlled through the compiler; when writing assembly language makes sense

Learning outcomes:

1. Understand how high-level language programs convert into executable code.
2. Know the capabilities and limits of compilers.
3. Comprehend basic representations of programs used to manipulate programs either in a compiler or by hand.

CE-ESY3 Real-time operating systems [core]
Minimum core coverage time: 3 hours

Topics:

- Context switching mechanisms
- Scheduling policies
- Rate-monotonic scheduling: theory and practice
- Priority inversion
- Other scheduling policies such as EDF
- Message-passing vs. shared memory communication
- Interprocess communication styles such as mailbox and RPC

Learning outcomes:

1. Distinguish RTOSs from workstation/server OS.
2. Distinguish real-time scheduling from traditional OS scheduling.
3. Understand major real-time scheduling policies.
4. Understand interprocess communication mechanisms.

CE-ESY4 Low-power computing [core]
Minimum core coverage time: 2 hours

Topics:

- Sources of energy consumption: toggling, leakage
- Instruction-level strategies for power management: function unit management
- Memory system power consumption: caches, off-chip memory
- Power consumption with multiple processes
- System-level power management: deterministic, probabilistic methods

Learning outcomes:

1. Understand why low-power computing is important.
2. Identify sources of energy consumption.
3. Identify possible remedies for energy consumption at various levels of design abstraction.

CE-ESY5 Reliable system design [core]
Minimum core coverage time: 2 hours

Topics:

- Transient vs. permanent failures in hardware
- Sources of errors from software
- The role of design verification in reliable system design
- Fault-tolerance techniques
- Famous failures of embedded computers

Learning outcomes:

1. Understand the variety of sources of faults in embedded computing systems.
2. Identify strategies to find problems.
3. Identify strategies to minimize the effects of problems.

CE-ESY6 Design methodologies [core]
Minimum core coverage time: 3 hours

Topics:

- Multi-person design projects
- Designing on-time and on-budget
- Design reviews
- Tracking error rates and sources
- Change management

Learning outcomes:

1. Understand why real-world projects are not the same as class projects.
2. Identify important goals of the methodology.
3. Understand the importance of design tracking and documentation.

CE-ESY7 Tool support [elective]

Topics:

- Compilers and programming environments
- Logic analyzers
- RTOS tools
- Power analysis
- Software management tools
- Project management tools

Learning outcomes:

1. Understand role of hardware and software tools in system development.
2. Understand how to use tools to support the methodology.

CE-ESY8 Embedded multiprocessors [elective]

Topics:

- Importance of multiprocessors as in performance, power, and cost
- Hardware/software partitioning for single-bus systems
- More general architectures
- Platform FPGAs as multiprocessors

Learning outcomes:

1. Understand the use of multiple processors in embedded systems.
2. Identify trade-offs between CPUs and hardwired logic in multiprocessors.
3. Understand basic design techniques.

CE-ESY9 Networked embedded systems [elective]

Topics:

- Why networked embedded systems
- Example networked embedded systems: automobiles, factory automation systems
- The OSI reference model
- Types of network fabrics
- Network performance analysis
- Basic principles of the Internet protocol
- Internet-enabled embedded systems

Learning outcomes:

1. Understand why networks are components of embedded systems.
2. Identify roles of hardware and software in networked embedded systems.
3. Compare networks designed for embedded computing with Internet networking.

CE-ESY10 Interfacing and mixed-signal systems [elective]

Topics:

- Digital-to-analog conversion
- Analog-to-digital conversion
- How to partition analog/digital processing in interfaces
- Digital processing and real-time considerations

Learning outcomes:

1. Understand pros and cons of digital and analog processing in interfaces.
2. Understand fundamentals of A/D and D/A conversion.

Human-Computer Interaction (CE – HCI)

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

* Consult the CC2001 Report [ACM/IEEECS, 2001] HC Knowledge Area for more detail

CE-HCI0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying human-computer interaction
- Highlight some people that influenced or contributed to the area of human-computer interaction
- Indicate some important HCI considerations such as foundational elements, ergonomic designs, and graphical interfaces
- Contrast ways in which engineering design should reflect human interaction
- Mention some advantages of small-screen designs versus large-screen designs
- Describe how one would evaluate an engineering design vis-à-vis HCI compatibility
- Explore some additional resources associated with human-computer interaction
- Explain the purpose and role of human-computer interaction in computer engineering

Learning outcomes:
1. Identify some contributors to human-computer interaction and relate their achievements to the knowledge area.
2. Define HCI.
3. Explain the reasons for proper HCI designs in engineering.
4. Provide a good reason for having a small-screen graphical user interface.
5. Provide a good reason for having a large-screen graphical user interface.
6. Give an example on how one might evaluate an engineering design using some principles of HCI.
7. Describe how computer engineering uses or benefits from human-computer interaction.

CE-HCI1 Foundations of human-computer interaction [core]
Minimum core coverage time: 2 hours

Topics:
- Motivation: the importance of the human interface in computer engineering; issues of small screens and larger screens
- The range of possibilities: text-based systems, use of graphics, sound, animation, video; the possibilities of multimedia Strengths and weaknesses of individual approaches
- The web as an example of an interface
- Human-centered development and evaluation
- Human performance models: perception, movement, and cognition; culture, communication, and organizations
- Accommodating human diversity; the role of multimedia
- Principles of good human computer interaction design in the context of computer engineering; engineering tradeoffs
- Introduction to usability testing
- The role of and use of a range of tools

Learning outcomes:
1. Develop a conceptual vocabulary for analyzing human interaction with software: to include terms such as affordance, conceptual model, and feedback.
2. Summarize the basic science of psychological and social interaction relevant to the development of human computer interfaces.
3. Differentiate between the role of hypotheses and experimental results recognizing the role of correlations.
4. Distinguish between the different interpretations that a given icon, symbol, word, or color can have in (a) different human cultures and (b) in the context of human diversity.
5. Create and conduct a simple usability test for an existing software application, taking into account human diversity.

CE-HCI2 Graphical user interface [core]
Minimum core coverage time: 2 hours

Topics:
- Illustrations of developments of graphical user interfaces including: textual displays; interfaces that include alarms; displays that exhibit motion; displays that exhibit interaction
- Principles of design using graphical user interfaces (GUIs); principles associated with interaction including fault tolerance
- GUI toolkits
- Principles associated with use of sound and multimedia in different contexts; use of relevant tools
- Principles of design for web interfaces; web interfaces for small screen and mobile devices
- Use of relevant tools

Learning outcomes:
1. Identify several fundamental principles for effective GUI design relevant for different applications in computer engineering.
2. Use a GUI toolkit to create a simple application that supports a graphical user interface.
3. Illustrate the effect of fundamental design principles on the structure of a graphical user interface.
4. Conduct a simple usability test for each instance and compare the results.

CE-HCI3 I/O technologies [core]
Minimum core coverage time: 1 hour

Topics:
- The range of technologies and techniques that can be deployed in intelligent systems: vision, speech processing, specialized sensors
- Technologies for location aware computing, the role of geographical positioning systems, other possibilities
- Overview of the technologies involved: their strengths and the limitations
- Availability of software support and of relevant tools

Learning outcomes:
1. Recognize contexts in which to deploy the various technologies associated with intelligent systems.
2. Demonstrate an awareness of the capabilities as well as the limitations of the available techniques and technologies.

CE-HCI4 Intelligent systems [core]
Minimum core coverage time: 2 hours

Topics:
- Illustrations of the deployment of intelligent systems in a computer engineering context
- The nature of intelligence deployed and the implications for sensors, for software (the nature of the software, the reliability of the software, the reasoning, the speed of response)
- The special case of mobile systems and location aware devices; illustrations of applications and benefits
- The problems associated with control passing to an agent and a user losing control as in a safety context
- Ethical issues
- History of artificial intelligence
- Philosophical questions about the nature of intelligence: the Turing test; Searle's "Chinese Room" thought experiment
- Ethical issues in artificial intelligence; the concepts of the computable and the incomputable
- Fundamental definitions: Optimal vs. human-like reasoning, optimal vs. human-like behavior
- The nature of knowledge and knowledge based systems; consistency and completeness concerns; what is possible
- The issues associated with the ordering of information, the fundamental role of technologies such as search, inference, and the role of heuristics
- Modeling the world; guidance on effective approaches

Learning outcomes:
1. For a range of contexts in which intelligent systems are deployed in a computer engineering context, identify the technical implications for devices, for computing power and for software
2. Identify the potential for the use of intelligent systems in a range of computer engineering equipment
3. Discuss the professional, legal and ethical implications of deploying intelligent systems in a range of computer engineering situations
4. Describe situations from computer engineering applications when intelligent systems can be relied upon to deliver a required response
5. Describe situations in which intelligent systems may or may not be reliable enough to deliver a required response, giving reasons for the answer.
6. Explain the necessity for heuristics in the general context of intelligent systems
7. Differentiate between the concepts of: optimal reasoning and human-like reasoning; of optimal behavior and human-like behavior.

CE-HCI5 Human-centered software evaluation [elective]

Topics:

- Setting goals for evaluation
- The range of evaluation criteria including learning time, task time, completion time, acceptability; the strengths and weaknesses of the different criteria
- Evaluation without users: walkthroughs, Keystroke Level Model (KLM), guidelines, and standards
- Evaluation with users: usability testing, interview, survey, experiment

Learning outcomes:

1. Discuss the full range of evaluation criteria appropriate for one of a range of computer engineering applications.
2. Conduct a walkthrough and a Keystroke Level Model analysis.
3. Summarize the features of the major guidelines and standards associated with human-centered software evaluation
4. Evaluate one of a range of existing interactive system with appropriate human-centered criteria and usability, giving reasons for selection of techniques.

CE-HCI6 Human-centered software development [elective]

Topics:

- General guidelines associated with the structure of large systems that embody significant HCI code; separation of concerns, issues of maintenance; differing development life cycles
- Approaches, characteristics, and overview of processes associated with human centered software; systems that offer interfaces in different natural languages
- Functionality and usability: task analysis, interviews, surveys
- Specifying presentation and interaction; techniques and approaches; software support
- Prototyping techniques and tools: Paper storyboards; inheritance and dynamic dispatch; prototyping languages and GUI builders
- Quality considerations
- Standards and guidelines

Learning outcomes:

1. Explain the basic types and features of human-centered software development.
2. Indicate three functional and three usability requirements that may be useful in developing human-centered software.
3. Specify an interactive object using one of the common methods as well as appropriate standards or guidelines.
4. Demonstrate the application of guidelines and fundamental principles in developing one of a range of possible computer engineering applications that rely on a human computer interface.

CE-HCI7 Interactive graphical user-interface design [elective]

Topics:

- Choosing interaction styles and interaction techniques appropriate to applications
- HCI aspects of common widgets
- HCI aspects of screen design: layout, color, fonts, and labeling
- Handling human failure
- Beyond simple screen design: visualization, representation, metaphor
- Interfaces for computer engineering tools that utilize databases
- Multi-modal interaction: graphics, sound, and haptics
- 3D interaction and virtual reality

Learning outcomes:

1. Summarize common interaction styles.
2. Explain good design principles of each of the following: common widgets; sequenced screen presentations; simple error-trap dialog; a user manual.
3. Design, prototype, and evaluate a simple 2D GUI
4. Discuss the challenges that exist in moving from 2D to 3D interaction.

CE-HCI8 Graphical user-interface programming [elective]

Topics:

- User interface management systems: different approaches; responsibilities of the application and of the application
- Kernel based and client server models for the user interface
- Dialogue independence and levels of analysis, Seeheim model
- Widget classes; aggregation of widgets
- Event management and user interaction
- Geometry management, constraint based approaches
- GUI builders and user interface programming environments; callbacks and their role in GUI builders
- Cross-platform design

Learning outcomes:

1. Compare the event-driven paradigm with more traditional procedural control for the user interface
2. Identify common differences as well as similarities in cross-platform user interface design.
3. Demonstrate an ability to outline an approach to interface design for a computer engineering application that utilizes an appropriately chosen selection of technologies from event management, widgets, geometry management, and GUI builders.

CE-HCI9 Graphics and visualization [elective]

Topics:

- Computer graphics: Design of models that represent information and support the creation and viewing of images; possibilities to include two dimensions, three dimensions, shading, animation; graphical display devices; packages that support graphical design
- Visualization: Nature of computer visualization; the role of visualization in communicating information in a dataset to an interested party; use of tools to accomplish this
- Virtual reality: Nature of and benefit of virtual reality; its limitations; components of a typical virtual reality situation, e.g. graphics, sound; the nature of interaction with a user; virtual environments
- Computer vision: Role in deducing properties and the structure of a three dimensional world from one or more two dimensional images; tools used for this and their role in computer engineering

Learning outcomes:

1. Understand the nature of graphical design and implement a simple graphical activity using a standard software package.
2. Appreciate the role of visualization technologies and demonstrate them through the development of a simple application.
3. Appreciate the benefits of virtual reality and the nature of the advantages this offers.
4. Demonstrate a simple application of computer vision technology in a computer engineering context.

CE-HCI10 Multimedia systems [elective]

Topics:

- The use of multi-media in computer engineering applications; the benefit, especially in reinforcement and in a context of human diversity
- The implications of performance requirements associated with multi-media for the hardware, the software and communications aspects of computer based systems
- Considerations resulting from interaction of various kinds
- Design concerns associated with the development of multi-media interfaces
- Implementation issues; synchronization aspects, tools
- Quality considerations
- Guidelines and standards

Learning outcomes:

1. Select system components which are suitable for the realization of multi-media interfaces of high quality
2. Design and develop a multi-media interface for a simple computer engineering application.

Computer Networks (CE-NWK)

CE-NWK0 History and overview [core]
CE-NWK1 Communications network architecture [core]
CE-NWK2 Communications network protocols [core]
CE-NWK3 Local and wide area networks [core]
CE-NWK4 Client-server computing [core]
CE-NWK5 Data security and integrity [core]
CE-NWK8 Wireless and mobile computing [core]
CE-NWK6 Performance evaluation [elective]
CE-NWK7 Data communications [elective]
CE-NWK9 Network management [elective]
CE-NWK10 Compression and decompression [elective]

CE-NWK0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying networks
- Highlight some people that influenced or contributed to the area of networks
- Indicate some important topic areas such as network architectures and protocols, network types (LAN, WAN, MAN, and wireless), data security, data integrity, and network performance
- Describe some of the hardware and software components of networks
- Describe the operation of some network devices such as repeaters, bridges, switches, routers, and gateways
- Indicate some network topologies such as mesh, star, tree, bus, and ring
- Describe the purpose of network protocols
- Mention some popular protocols
- Explore some additional resources associated with networks
- Explain the purpose and role of networks in computer engineering

Learning outcomes:
1. Identify some contributors to networks and relate their achievements to the knowledge area.
2. Identify some components of a network.
3. Name some network devices and describe their purpose.
4. Describe advantages of a star topology over a ring topology.
5. Describe advantages of a ring topology over a star topology.
6. Define the meaning of a protocol.
7. Explain the importance of security when dealing with networks.
8. Describe how computer engineering uses or benefits from networks.

CE-NWK1 Communications network architecture [core]
Minimum core coverage time: 3 hours

Topics:
- Network line configuration (point-to-point, multipoint)
- Networking and internetworking devices: Repeaters, bridges, switches, routers, gateways
- Network Topologies (mesh, star, tree, bus, ring)
- Connection-oriented and connectionless services

Learning outcomes:
1. Understand fundamental concepts of networks and their topologies.
2. Understand the concept of network architecture and its hardware components.

CE-NWK2 Communications network protocols [core]
Minimum core coverage time: 4 hours

Topics:
- Network protocol (syntax, semantics, timing)
- Protocol suites (TCP/IP)
- Layered protocol software (stacks): Physical layer networking concepts; data link layer concepts; internetworking and routing
- Network Standards and standardization bodies

Learning outcomes:
1. Demonstrate understanding of the elements of a protocol, and the concept of layering.
2. Recognize the importance of networking standards, and their regulatory committees.
3. Describe the seven layers of the OSI model.
4. Compare and contrast the OSI model with the TCP/IP model.
5. Demonstrate understanding of the differences between circuit switching and packet switching.

CE-NWK3 Local and wide area networks [core]
Minimum core coverage time: 4 hours

Topics:
- LAN topologies (bus, ring, star)
- LAN technologies (Ethernet, token Ring, Gigabit Ethernet)
- Error detection and correction
- Carrier sense multiple access networks (CSMA)
- Large networks and wide areas
- Circuit switching and packet switching
- Protocols (addressing, congestion control, virtual circuits, quality of service)

Learning outcomes:
1. Understand the basic concepts of LAN and WAN technologies and topologies.
2. Demonstrate understanding of different components and requirements of network protocols.
3. Demonstrate understanding of basic concepts of error detection and correction at the data link layer and below.
4. Design and build a simple network by implementing (and designing) a simple network protocol that operates at the physical and data link layers of the OSI model.

CE-NWK4 Client-server computing [core]
Minimum core coverage time: 3 hours

Topics:
- Web technologies: Server-side programs; common gateway interface (CGI) programs; client-side scripts; The applet concept
- Characteristics of web servers: Handling permissions; file management; capabilities of common server architectures
- Support tools for web site creation and web management

Learning outcomes:
1. Explain the different roles and responsibilities of clients and servers for a range of possible applications.
2. Select a range of tools that will ensure an efficient approach to implementing various client-server possibilities.
3. Design and build a simple interactive web-based application (e.g., a simple web form that collects information from the client and stores it in a file on the server).

CE-NWK5 Data security and integrity [core]
Minimum core coverage time: 4 hours

Topics:
- Fundamentals of secure networks; cryptography
- Encryption and privacy: Public key, private key, symmetric key
- Authentication protocols
- Packet filtering
- Firewalls
- Virtual private networks
- Transport layer security

Learning outcomes:

1. Understand common barriers to network security and the major issues involved in implementing proper security measures.
2. Describe the purpose of encryption and the function of public and private keys.
3. Compare and contrast the various types of firewalls.
4. Generate and distribute a PGP key pair and use the PGP package to send an encrypted e-mail message.
5. Explain the concept of and necessity for transport layer security.

CE-NWK6 Wireless and mobile computing [core]
Minimum core coverage time: 2 hours

Topics:

- Overview of the history, evolution, and compatibility of wireless standards
- The special problems of wireless and mobile computing
- Wireless local area networks and satellite-based networks
- Mobile Internet protocol
- Mobile aware adaptation
- Extending the client-server model to accommodate mobility
- Mobile data access: server data dissemination and client cache management
- The software packages to support mobile and wireless computing
- The role of middleware and support tools
- Performance issues
- Emerging technologies

Learning outcomes:

1. Describe the main characteristics of mobile IP and explain how differs from IP with regard to mobility management and location management as well as performance.
2. Illustrate (with home agents and foreign agents) how e-mail and other traffic is routed using mobile IP.
3. Be aware of the many areas of interest that lie within this area, including networking, multimedia, wireless, and mobile computing, and distributed computing.

CE-NWK7 Performance evaluation [elective]

Topics:

- Privacy and public networks
- Virtual private networks
- Service paradigms: connection-oriented service; connectionless service; network performance characteristics; delay, throughput

Learning outcomes:

1. Define performance metrics.
2. Describe how each affects a particular network and/or service paradigm.

CE-NWK8 Data communications [elective]

Topics:

- Encoding and modulating: A/D and D/A conversion
- Interfaces and modems
- Transmission media
- Multiplexing
- Error detection and correction

Learning outcomes:

1. Demonstrate understanding of the fundamental concepts of data communications.
2. Understand signals and signal encoding methods to communication service methods and data transmission modes.

CE-NWK9 Network management [elective]

Topics:

- Overview of the issues of network management
- Use of passwords and access control mechanisms
- Domain names and name services
- Issues for Internet service providers (ISPs)
- Security issues and firewalls
- Quality of service issues: performance, failure recovery

Learning outcomes:

1. Explain the issues for network management arising from a range of security threats, including viruses, worms, Trojan horses, and denial-of-service attacks
2. Summarize the strengths and weaknesses associated with different approaches to security.
3. Develop a strategy for ensuring appropriate levels of security in a system designed for a particular purpose.
4. Implement a network firewall.

CE-NWK10 Compression and decompression [elective]

Topics:

- Analog and digital representations
- Encoding and decoding algorithms
- Lossless and lossy compression
- Data compression: Huffman coding and the Ziv-Lempel algorithm
- Audio compression and decompression
- Image compression and decompression
- Video compression and decompression
- Performance issues: timing, compression factor, suitability for real-time use

Learning outcomes:

1. Summarize the basic characteristics of sampling and quantization for digital representation.
2. Select, giving reasons that are sensitive to the specific application and particular circumstances, the most appropriate compression techniques for text, audio, image, and video information.
3. Explain the asymmetric property of compression and decompression algorithms.
4. Illustrate the concept of run-length encoding.
5. Illustrate how a program like the UNIX *compress* utility, which uses Huffman coding and the Ziv-Lempel algorithm, would compress a typical text file.

Operating Systems (CE-OPS)

CE-OPS0	**History and overview [core]**
CE-OPS1	**Design principles [core] ***
CE-OPS2	**Concurrency [core] ***
CE-OPS3	**Scheduling and dispatch [core] ***
CE-OPS4	**Memory management [core] ***
CE-OPS5	**Device management [elective] ***
CE-OPS6	**Security and protection [elective] ***
CE-OPS7	**File systems [elective] ***
CE-OPS8	**System performance evaluation [elective] ***

* Consult the CC2001 Report [ACM/IEEECS, 2001] OS Knowledge Area for more detail

CE-OPS0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying operating systems
- Highlight some people that influenced or contributed to the area of operating systems
- Indicate some important topic areas such as function and design, concurrency, scheduling, dispatch, memory management, device management, file systems, security, and protection
- Describe the purpose of an operating system
- Indicate the meaning of an interrupt
- Describe the meaning of concurrency and the reasons for its importance
- Illustrate the manner in which scheduling and dispatch take place in a computer through its operating system
- Describe the manner and importance of memory management
- Describe the manner and importance of device management
- Explore some additional resources associated with operating systems
- Explain the purpose and role of operating systems in computer engineering

Learning outcomes:
1. Identify some contributors to operating systems and relate their achievements to the knowledge area.
2. Provide some reasons for a computer to have an operating system.
3. Describe concurrency and reasons for its importance.
4. Describe scheduling and illustrate how it works to improve computer performance.
5. Sketch an example of how and why a compute would need to manage memory.
6. Identify some devices an operating system would manage.
7. Describe how computer engineering uses or benefits from operating systems.

CE-OPS1 Design principles [core]
Minimum core coverage time: 5 hours

Topics:
- Functionality of a typical operating system
- Mechanisms to support client-server models, hand-held devices
- Design issues (efficiency, robustness, flexibility, portability, security, compatibility)
- Influences of security, networking, multimedia, windows
- Structuring methods (monolithic, layered, modular, micro-kernel models)
- Abstractions, processes, and resources
- Concepts of application program interfaces (APIs) specific to operating systems
- Applications needs and the evolution of hardware/software techniques
- Device organization
- Interrupts: methods and implementations
- Concept of user/system state and protection, transition to kernel mode

Learning outcomes:
1. Demonstrate understanding of Operating Systems as an interface between user programs and the computer hardware.
2. Demonstrate understanding of the logical layers and the benefits of building these layers in a hierarchical fashion.
3. Relate system state to user protection.
4. Explain the range of requirements that a modern operating system has to address.
5. Define the functionality that a modern operating system must deliver to meet a particular need.
6. Articulate design tradeoffs inherent in operating system design.

CE-OPS2 Concurrency [core]
Minimum core coverage time: 6 hours

Topics:
- States and state diagrams
- Structures (ready list, process control blocks, and so forth)
- Dispatching and context switching
- The role of interrupts
- Concurrent execution: advantages and disadvantages
- The "mutual exclusion" problem and some solutions
- Deadlock: causes, conditions, prevention
- Models and mechanisms (semaphores, monitors, condition variables, rendezvous)
- Producer-consumer problems and synchronization
- Multiprocessor issues (spin-locks, reentrancy)

Learning outcomes:
1. Justify the presence of concurrency within the framework of an operating system.
2. Demonstrate the potential run-time problems arising from the concurrent operation of many (possibly a dynamic number of) tasks.
3. Summarize the range of mechanisms (at an operating system level) that are useful to realize concurrent systems and be able to describe the benefits of each.
4. Explain the different states that a task may pass through and the data structures needed to support the management of many tasks.

CE-OPS3 Scheduling and dispatch [core]
Minimum core coverage time: 3 hours

Topics:
- Preemptive and non-preemptive scheduling
- Schedulers and policies
- Processes and threads
- Deadlines and real-time issues

Learning outcomes:
1. Compare and contrast the common algorithms used for both preemptive and non-preemptive scheduling of tasks in operating systems.
2. Describe relationships between scheduling algorithms and application domains.
3. Investigate the wider applicability of scheduling in such contexts as disk I/O, networking scheduling, and project scheduling.

CE-OPS4 Memory management [core]
Minimum core coverage time: 5 hours

Topics:
- Review of physical memory and memory management hardware
- Overlays, swapping, and partitions
- Paging and segmentation
- Placement and replacement policies
- Working sets and thrashing
- Caching

Learning outcomes:
1. Introduce memory hierarchy and cost-performance tradeoffs.
2. Explain virtual memory and its realization in hardware and software.
3. Examine the wider applicability and relevance of the concepts of virtual entity and of caching.
4. Evaluate the trade-offs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed.
5. Defend the different ways of allocating memory to tasks based on the relative merits of each.

CE-OPS5 Device management [elective]

Topics:

- Characteristics of serial and parallel devices
- Abstracting device differences
- Buffering strategies
- Direct memory access
- Recovery from failures

Learning outcomes:

1. Identify the relationship between the physical hardware and the virtual devices maintained by the operating system.
2. Differentiate the mechanisms used in interfacing a range of devices (including hand-held devices, networks, multimedia) to a computer and explain the implications of these for the design of an operating system.
3. Implement a simple device driver for a range of possible devices.

CE-OPS6 Security and protection [elective]

Topics:

- Overview of system security
- Policy/mechanism separation
- Security methods and devices
- Protection, access, and authentication
- Models of protection
- Memory protection
- Encryption
- Recovery management

Learning outcomes:

1. Defend the need for protection and security, and the role of ethical considerations in computer use.
2. Summarize the features of an operating system used to provide protection and security, and describe the limitations of each of these.
3. Compare and contrast current methods for implementing security.

CE-OPS7 File systems [elective]

Topics:

- Files: data, metadata, operations, organization, buffering, sequential, nonsequential
- Directories: contents and structure
- File systems: partitioning, mount/unmount, and virtual file systems
- Standard implementation techniques
- Memory-mapped files
- Special-purpose file systems
- Naming, searching, access, backups

Learning outcomes:

1. Summarize the full range of considerations that support file systems.
2. Compare and contrast different approaches to file organization, recognizing the strengths and weaknesses of each.

CE-OPS8 System performance evaluation [elective]

Topics:

- Why and what system performance needs to be evaluated
- Policies for caching, paging, scheduling, memory management, security, and so forth
- Evaluation models: deterministic, analytic, simulation, or implementation-specific
- How to collect evaluation data (profiling and tracing mechanisms)

Learning outcomes:

1. Describe the performance metrics used to determine how a system performs.
2. Explain the main evaluation models used to evaluate a system.

Programming Fundamentals (CE-PRF)

CE-PRF0 History and overview [core]
CE-PRF1 Programming paradigms [core] *
CE-PRF2 Programming constructs [core] *
CE-PRF3 Algorithms and problem-solving [core] *
CE-PRF4 Data structures [core] *
CE-PRF5 Recursion [core] *
CE-PRF6 Object-oriented programming [elective] *
CE-PRF7 Event-driven and concurrent programming [elective] *
CE-PRF8 Using APIs [elective] *

* Consult the CC2001 Report [ACM/IEEECS, 2001] PF Knowledge Area for more detail ail

CE-PRF0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying programming fundamentals
- Highlight some people that influenced or contributed to the area of programming fundamentals
- Indicate some important topic areas such as programming constructs, algorithms, problem solving, data structures, programming paradigms, recursion, object-oriented programming, event-driven programming, and concurrent programming
- Contrast between an algorithm and a data structure
- Distinguish between a variable, type, expression, and assignment
- Highlight the role of algorithms in solving problems
- Describe some of the fundamental data structures such as array, record, stack, and queue
- Explain the various programming paradigms such as procedural, functional, logic, and object oriented
- Explain how divide-and-conquer strategies lend themselves to recursion
- Explore some additional resources associated with programming fundamentals
- Explain the purpose and role of programming fundamentals in computer engineering

Learning outcomes:
1. Identify some contributors to programming fundamentals and relate their achievements to the knowledge area.
2. Define the meaning of algorithm and data structure.
3. Know the reasons that a way to solve problems is by using algorithms.
4. Distinguish the difference between a stack and a queue.
5. Identify the difference between various programming paradigms.
6. Explain recursion and the way it works.
7. Describe how computer engineering uses or benefits from programming fundamentals.

CE-PRF1 Programming paradigms [core]
Minimum core coverage time: 5 hours

Topics:
- Procedural programming
- Functional programming
- Object-oriented design
- Encapsulation and information-hiding
- Separation of behavior and implementation
- Classes, subclasses, and inheritance
- Event-Driven programming

Learning outcomes:
1. Identify the paradigm used by pseudo-code snippets.
2. Identify the appropriate paradigm for a given programming problem.

CE-PRF2 Programming constructs [core]
Minimum core coverage time: 7 hours

Topics:
- Basic syntax and semantics of a high-level language
- Variables, types, expressions, and assignment
- Simple I/O
- Conditional and iterative control structures
- Functions and parameter passing

Learning outcomes:
1. Analyze and explain the behavior of simple programs involving the fundamental programming constructs covered by this unit.
2. Write a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of procedures and functions.

CE-PRF3 Algorithms and problem-solving [core]
Minimum core coverage time: 8 hours

Topics:
- Problem-solving strategies
- The role of algorithms in the problem-solving process
- Implementation strategies for algorithms
- Debugging strategies
- The concept and properties of algorithms
- Structured decomposition

Learning outcomes:
1. Define the basic properties of an algorithm.
2. Develop algorithms for solving simple problems.
3. Use a suitable programming language to implement, test, and debug algorithms for solving simple problems.
4. Apply the techniques of structured decomposition to break a program into smaller pieces.

CE-PRF4 Data structures [core]
Minimum core coverage time: 13 hours

Topics:
- Primitive types
- Arrays
- Records
- Strings and string processing
- Data representation in memory
- Static, stack, and heap allocation
- Runtime storage management
- Pointers and references
- Linked structures
- Implementation strategies for stacks, queues, and hash tables
- Implementation strategies for graphs and trees
- Strategies for choosing the right data structure

Learning outcomes:
1. Identify data structures useful to represent specific types of information and discuss the tradeoffs among the different possibilities.
2. Write programs that use each of the following data structures: arrays, records, strings, linked lists, stacks, queues, and hash tables.
3. Describe the way a computer allocates and represents these data structures in memory.

CE-PRF5 Recursion [core]
Minimum core coverage time: 5 hours

Topics:
- The concept of recursion
- Recursive mathematical functions
- Divide-and-conquer strategies
- Recursive backtracking
- Implementation of recursion

Learning outcomes:
1. Explain the concept of recursion.
2. Explain the structure of the divide-and-conquer approach.
3. Write, test, and debug simple recursive functions and procedures.
4. Describe how recursion can be implemented using a stack.

CE-PRF6 Object-oriented programming [elective]

Topics:
- Polymorphism
- Class hierarchies
- Collection classes and iteration protocols
- Fundamental design patterns

Learning outcomes:
1. Outline the philosophy of object-oriented design and the concepts of encapsulation, subclassing, inheritance, and polymorphism.
2. Design, code, test, and debug simple programs in an object-oriented programming language.
3. Select and apply appropriate design patterns in the construction of an object-oriented application.

CE-PRF7 Event-driven and concurrent programming [elective]

Topics:
- Event-handling methods
- Event propagation
- Managing concurrency in event handling
- Exception handling

Learning outcomes:
1. Design, code, test, and debug simple event-driven programs that respond to user events.
2. Defend the need for concurrency control and describe at least one method for implementing it.
3. Develop code that responds to exception conditions raised during execution.

CE-PRF8 Using APIs [elective]

Topics:
- API programming
- Class browsers and related tools
- Programming by example
- Debugging in the API environment
- Component-based computing
- Middleware

Learning outcomes:
1. Explain the value of application programming interfaces (APIs) in software development.
2. Design, write, test, and debug programs that use large-scale API packages.

Probability and Statistics (CE-PRS)

CE-PRS0 History and overview [core]
CE-PRS1 Discrete probability [core]
CE-PRS2 Continuous probability [core]
CE-PRS3 Expectation [core]
CE-PRS4 Stochastic processes [core]
CE-PRS5 Sampling distributions [core]
CE-PRS6 Estimation [core]
CE-PRS7 Hypothesis tests [core]
CE-PRS8 Correlation and regression [elective]

CE-PRS0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying probability and statistics
- Highlight some people that influenced or contributed to the area of probability and statistics
- Indicate some important topic areas such as discrete probability, continuous probability, expectation, sampling, estimations, stochastic process, correlation, and regression
- Describe the meaning of discrete probability
- Describe the meaning of continuous probability
- Contrast discrete from continuous probability
- Provide a context for considering probabilistic expectation
- Indicate the reason for using sampling distributions
- Define a stochastic process
- Mention the need for considering stochastic processes
- Describe the need for probabilistic estimation in computer engineering
- Highlight the importance of correlation
- Provide examples for using regression
- Explore some additional resources associated with probability and statistics
- Explain the purpose and role of probability and statistics in computer engineering

Learning outcomes:
1. Identify some contributors to probability and statistics and relate their achievements to the knowledge area.
2. Contrast the difference between probability and statistics.
3. Give some examples for using probability and statistics.
4. Contrast the difference between discrete and continuous probability.
5. Identify some discrete and continuous probability distributions.
6. Articulate the importance of estimation.
7. Identify the meaning of correlation.
8. Identify the meaning of regression.
9. Describe how computer engineering uses or benefits from probability and statistics.

CE-PRS1 Discrete probability [core]
Minimum core coverage time: 6 hours

Topics:
- Randomness, finite probability space, probability measure, events
- Conditional probability, independence, Bayes' theorem
- Discrete random variables
- Binomial, Poisson, geometric distributions
- Mean and variance: concepts, significance, computations, applications
- Integer random variables

Learning outcomes:
1. Calculate probabilities of events and expectations of random variables for elementary problems such as games of chance.
2. Differentiate between dependent and independent events.
3. Apply binomial theorem to independent events and Bayes' theorem to dependent events.
4. Apply the tools of probability to solve problems such as in the Monte Carlo method, the average case analysis of algorithms and hashing.

CE-PRS2 Continuous probability [core]
Minimum core coverage time: 6 hours

Topics:
- Continuous random variables, the nature of these, illustrations of use
- Exponential and normal distribution: probability density functions, calculation of mean and variance
- The central limit theorem and the implications for the normal distribution
- Joint distribution

Learning outcomes:
1. Recognize situations in which it is appropriate to consider the relevance of the normal distribution and/or the exponential distribution.
2. Calculate the mean and the variance for given distributions involving continuous random variables.

CE-PRS3 Expectation [core]
Minimum core coverage time: 4 hours

Topics:
- Moments, transform methods, mean time to failure
- Conditional expectation, examples
- Imperfect fault coverage and reliability

Learning outcomes:
1. Understand the significance and be able to compute expectation of functions of more than one variable and transform.
2. Compute fault coverage and reliability in simple hardware and software applications.

CE-PRS4 Stochastic processes [core]
Minimum core coverage time: 6 hours

Topics:
- Introduction: Bernoulli and Poisson processes, renewal process, renewal model of program behavior
- Discrete parameter Markov chains: transition probabilities, limiting distributions
- Queuing: M/M1 and M/G/1, birth and death process
- Finite Markov chains, program execution times

Learning outcomes:
1. Become familiar with the concepts and tools to manipulate stochastic processes.
2. Apply the concepts and tools of stochastic processes to analyze the performance of simple hardware and software systems.

CE-PRS5 Sampling distributions [core]
Minimum core coverage time: 4 hours

Topics:
- Purpose and the nature of sampling, its uses and applications
- Random approaches to sampling: basic method, stratified sampling and variants thereof, cluster sampling
- Non-random approaches: purposive methods, sequential sampling
- Data analysis; tools; graphical and numerical summaries
- Multivariate distributions, independent random variables

Learning outcomes:
1. Recognize situations in which the different approaches to sampling are relevant.
2. Demonstrate the ability to apply appropriate sampling methods in a range of situations.

CE-PRS6 Estimation [core]
Minimum core coverage time: 4 hours

Topics:
- Nature of estimates: point estimates, interval estimates
- Criteria to be applied to single point estimators: unbiased estimators, consistent estimators, efficiency and sufficiency of estimators
- Maximum likelihood principle approach, least squares approach; applicability conditions for these
- Confidence intervals
- Estimates for one or two samples

Learning outcomes:
1. Describe the fundamental principles behind the concept of estimation and give examples that illustrate its beneficial application.
2. Given a distribution, apply basic principles to derive estimators that exhibit desirable properties.

CE-PRS7 Hypothesis tests [core]
Minimum core coverage time: 2 hours

Topics:
- Development of models and associated hypotheses, the nature of these
- Hypothesis formulation: null and alternate hypotheses
- Testing hypothesis based on a single parameter, choice of test statistic; choice of samples and distributions
- Criteria for acceptance of hypothesis
- t-test, chi-squared test; applicability criteria for these

Learning outcomes:
1. Explain the role of hypothesis testing, describing the main steps in the process.
2. Given a sample situation, formulate a hypothesis and carry out appropriate tests to check its acceptability.

CE-PRS8 Correlation and regression [elective]

Topics:
- The nature of correlation and regression, definitions
- Definition and calculation of correlation coefficients
- Approaches to correlation: the linear model approach, the least squares fitting approach, strengths and weaknesses of these and conditions for applicability

Learning outcomes:
1. Recognize circumstances under which it is appropriate to investigate relationships between variables.
2. Given a suitable circumstance, apply correlation and regression techniques with a view to establishing relationships between variables.

Social and Professional Issues (CE-SPR)

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

* Consult the CC2001 Report [ACM/IEEECS, 2001] SP Knowledge Area for more detail

CE-SPR0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying social and professional issues
- Highlight some people that influenced or contributed to the area of social and professional issues
- Indicate some important topic areas such social context of computing, professional and ethical responsibilities, risks and trade-offs, intellectual property, privacy, and codes of ethics and professional conduct
- Contrast between what is legal to what is ethical
- Explain the importance of ethical integrity in the practice of computer engineering
- Mention some ways a computer engineer may have to make conflicting ethical choices in practicing the engineering profession
- Explain the meaning of whistle blowing and the dilemma it sometimes places on computer engineers
- Explain professionalism relative to a practicing engineer
- Show that credentialing preserves the integrity of a professional
- Describe risk and its contrast with safety
- Explain the difference between a patent and a copyright
- Describe how privacy issues affect the practice of computer engineering
- Explore some additional resources associated with social and professional issues
- Explain the purpose and role of social and professional issues in computer engineering

Learning outcomes:
1. Identify some contributors to social and professional issues and relate their achievements to the knowledge area.
2. Contrast between ethical and legal issues.
3. Contrast between a patent and a copyright.
4. Identify some ways of credentialing a person to practice computer engineering.
5. Describe issues that contrast risk issues with safety issues.
6. Identify some issues in computer engineering that address privacy.
7. Describe whistle blowing and the conflicts between ethics and practice that may result from doing so.
8. Describe how computer engineering uses or benefits from social and professional issues.

CE-SPR1 Public policy [core]
Minimum core coverage time: 2 hours

Topics:
- Introduction to the social implications of computing
- Social implications of networked communication
- Growth of, control of, and access to the Internet
- Gender-related issues
- International issues

Learning outcomes:
1. Interpret the social context of a particular implementation.
2. Identify assumptions and values embedded in a particular design.
3. Evaluate a particular implementation using empirical data.
4. Describe positive and negative ways in which computing altars the modes of interaction between people.
5. Explain why computing/network access is restricted in some countries.

CE-SPR2 Methods and tools of analysis [core]
Minimum core coverage time: 2 hours

Topics:
- Making and evaluating ethical arguments
- Identifying and evaluating ethical choices
- Understanding the social context of design
- Identifying assumptions and values

Learning outcomes:
1. Analyze an argument to identify premises and conclusion.
2. Illustrate the use of example, analogy, and counter-analogy in ethical argument.
3. Detect use of basic logical fallacies in an argument.
4. Identify stakeholders in an issue and our obligations to them.
5. Articulate the ethical tradeoffs in a technical decision.

CE-SPR3 Professional and ethical responsibilities [core]
Minimum core coverage time: 2 hours

Topics:
- Community values and the laws by which we live
- The nature of professionalism
- Various forms of professional credentialing and the advantages and disadvantages
- The role of the professional in public policy
- The role of licensure and practice in engineering
- Contrasts of licensure in engineering but not other disciplines
- Maintaining awareness of consequences
- Ethical dissent and whistle blowing
- Codes of ethics, conduct, and practice (NSPE, IEEE, ACM, SE, AITP, and so forth)
- Dealing with harassment and discrimination
- "Acceptable use" policies for computing in the workplace

Learning outcomes:
1. Identify progressive stages in a whistle-blowing incident.
2. Specify the strengths and weaknesses of relevant professional codes as expressions of professionalism and guides to decision-making.
3. Provide arguments for and against licensure in non-engineering professions.
4. Identify ethical issues that arise in software development and determine how to address them technically and ethically.
5. Develop a computer use policy with enforcement measures.

CE-SPR4 Risks and liabilities [core]
Minimum core coverage time: 2 hours

Topics:
- Historical examples of software risks such as the Therac-25 case
- Product safety and public consumption
- Implications of software complexity
- Risk assessment and management

Learning outcomes:
1. Explain the limitations of testing as a means to ensure correctness.
2. Recognize the importance of product safety when designing computer systems.
3. Describe the differences between correctness, reliability, and safety.
4. Recognize unwarranted assumptions of statistical independence of errors.
5. Discuss the potential for hidden problems in reuse of existing components.

CE-SPR5 Intellectual property [core]
Minimum core coverage time: 2 hours

Topics:
- Foundations of intellectual property
- Copyrights, patents, and trade secrets
- Software piracy
- Software patents
- Transnational issues concerning intellectual property

Learning outcomes:
1. Distinguish among patent, copyright, and trade secret protection.
2. Discuss the legal background of copyright in national and international law.
3. Explain how patent and copyright laws may vary internationally.
4. Outline the historical development of software patents.

CE-SPR6 Privacy and civil liberties [core]
Minimum core coverage time: 2 hours

Topics:
- Ethical and legal basis for privacy protection
- Privacy implications of massive database systems
- Technological strategies for privacy protection
- Freedom of expression in cyberspace
- International and intercultural implications

Learning outcomes:
1. Summarize the legal bases for the right to privacy and freedom of expression in one's own nation.
2. Discuss how those concepts vary from country to country.
3. Describe current computer-based threats to privacy.
4. Explain how the Internet may change the historical balance in protecting freedom of expression.

CE-SPR7 Computer crime [core]
Minimum core coverage time: 1 hour

Topics:
- History and examples of computer crime
- "Cracking" ("hacking") and its effects
- Viruses, worms, and Trojan horses
- Crime prevention strategies

Learning outcomes:
1. Outline the technical basis of viruses and denial-of-service attacks.
2. Enumerate techniques to combat "cracker" attacks.
3. Discuss several different "cracker" approaches and motivations.
4. Identify the professional's role in security and the tradeoffs involved.

CE-SPR8 Economic issues in computing [core]
Minimum core coverage time: 2 hours

Topics:
- Costing out jobs with considerations on manufacturing, hardware, software, and engineering implications
- Cost estimates versus actual costs in relation to total costs
- Use of engineering economics in dealing with finances
- Entrepreneurship: prospects and pitfalls
- Monopolies and their economic implications
- Effect of skilled labor supply and demand on the quality of computing products
- Pricing strategies in the computing domain
- Differences in access to computing resources and the possible effects thereof

Learning outcomes:
1. Describe the assessment of total job costs.
2. Evaluate the risks of entering one's own business.
3. Apply engineering economic principles when considering fiscal arrangements.
4. Summarize the rationale for antimonopoly efforts.
5. Describe several ways in which shortages in the labor supply affect the information technology industry.
6. Suggest and defend ways to address limitations on access to computing.

CE-SPR9 Philosophical frameworks [elective]

Topics:
- Philosophical frameworks, particularly utilitarianism and deontological theories
- Problems of ethical relativism
- Scientific ethics in historical perspective
- Differences in scientific and philosophical approaches

Learning outcomes:
1. Summarize the basic concepts of relativism, utilitarianism, and deontological theories.
2. Recognize the distinction between ethical theory and professional ethics.
3. Identify the weaknesses of the "hired agent" approach, strict legalism, naïve egoism, and naïve relativism as ethical frameworks.

Software Engineering (CE – SWE)

CE-SWE0	**History and overview [core]**
CE-SWE1	**Software processes [core] ***
CE-SWE2	**Software requirements and specifications [core] ***
CE-SWE3	**Software design [core] ***
CE-SWE4	**Software testing and validation [core] ***
CE-SWE5	**Software evolution [core] ***
CE-SWE6	**Software tools and environments [core] ***
CE-SWE7	**Language translation [elective] ***
CE-SWE8	**Software project management [elective] ***
CE-SWE9	**Software fault tolerance [elective] ***

* Consult the CC2001 Report [ACM/IEEECS, 2001] SE Knowledge Area for more detail

CE-SWE0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying software engineering
- Highlight some people that influenced or contributed to the area of software engineering
- Indicate some important topic areas such as the software process, requirements, specifications, design, testing, validation, evolution, and project management
- Contrast software engineering with computer engineering
- Mention some examples that would use the software engineering approach
- Indicate the existence of formalized software processes such as the software life cycle
- Explain that requirements and specifications may change slightly as a software project evolves
- Indicate the importance of language selection when doing software design
- Highlight the importance of testing and validation in a software projects
- Explore some additional resources associated with software engineering
- Explain the purpose and role of software engineering in computer engineering

Learning outcomes:
1. Identify some contributors to software engineering and relate their achievements to the knowledge area.
2. Provide examples of the software process.
3. Articulate the difference between software engineering and computer engineering.
4. Articulate some of the components of a software process.
5. Provide some examples that would use software engineering.
6. Give reasons for the importance of testing and validation in the development of software.
7. Describe how computer engineering uses or benefits from software engineering.

CE-SWE1 Software processes [core]
Minimum core coverage time: 2 hours

Topics:
- Software life cycle and process models
- Process assessment models
- Software process metrics

Learning outcomes:
1. Select, with justification, the software development models most appropriate for the development and maintenance of diverse software products.
2. Explain the role of process maturity models.

CE-SWE2 Software requirements and specifications [core]
Minimum core coverage time: 2 hours

Topics:
- Requirements elicitation
- Requirements analysis modeling techniques
- Functional and nonfunctional requirements
- Prototyping
- Basic concepts of formal specification techniques

Learning outcomes:

1. Apply key elements and common methods for elicitation and analysis to produce a set of software requirements for a medium-sized software system.
2. Use a common, non-formal method to model and specify (in the form of a requirements specification document) the requirements for a medium-size software system (e.g., structured analysis or object-oriented-analysis).
3. Conduct a review of a software requirements document using best practices to determine the quality of the document.
4. Translate into natural language a software requirements specification written in a commonly used formal specification language.

CE-SWE3 Software design [core]
Minimum core coverage time: 2 hours

Topics:

- Fundamental design concepts and principles
- Software architecture
- Structured design
- Object-oriented analysis and design
- Component-level design
- Design for reuse

Learning outcomes:

1. Evaluate the quality of multiple software designs based on key design principles and concepts.
2. Using a software requirement specification and a common program design methodology and notation, create and specify the software design for a medium-size software product (e.g., using structured design or object-oriented design).
3. Using appropriate guidelines, conduct the review of a software design.

CE-SWE4 Software testing and validation [core]
Minimum core coverage time: 2 hours

Topics:

- Validation planning
- Testing fundamentals, including test plan creation and test case generation
- Black-box and white-box testing techniques
- Unit, integration, validation, and system testing
- Object-oriented testing
- Inspections

Learning outcomes:

1. Demonstrate the application of the different types and levels of testing (unit, integration, systems, and acceptance) to software products of medium size.
2. Undertake, as part of a team activity, an inspection of a medium-size code segment.
3. Describe the role that tools can play in the validation of software.

CE-SWE5 Software evolution [core]
Minimum core coverage time: 2 hours

Topics:

- Software maintenance: the different forms of maintenance; the associated disciples and the role and the nature of configuration management and version control
- Impact analysis; regression testing; associated software support
- Characteristics of maintainable software
- Software re-use in its different forms – their strengths and weaknesses
- Reengineering
- Legacy systems

Learning outcomes:

1. Identify the principal issues associated with software evolution and explain their impact on the software life cycle.
2. Develop a plan for re-engineering a medium-sized product in response to a change request.
3. Discuss the advantages and disadvantages of software reuse.
4. Demonstrate the ability to exploit opportunities for software reuse in a variety of contexts.

CE-SWE6 Software tools and environments [core]
Minimum core coverage time: 2 hours

Topics:
- Programming environments
- Requirements analysis and design modeling tools
- Testing tools
- Configuration management tools
- Tools based on databases – their design and development
- Additional possibilities including CASE tools
- Tool integration mechanisms

Learning outcomes:
1. Select, with justification, an appropriate set of tools to support the software development of a range of software products.
2. Analyze and evaluate a set of tools in a given area of software development (e.g., management, modeling, or testing).
3. Demonstrate the capability to use a range of software tools in support of the development of a software product of medium size.

CE-SWE7 Language translation [elective]

Topics:
- The range of tools that support software development for the computer engineer; the role of a formal semantics of a language
- Different possibilities regarding language translation: comparison of interpreters and compilers for high-level languages, and silicon compilers for hardware description languages, additional possibilities
- Language translation phases (lexical analysis, parsing, generation phase, optimization); separate compilation or translation - the benefits and the mechanisms; machine-dependent and machine-independent aspects of translation

Learning outcomes:
1. Compare and contrast compiled and interpreted execution models, outlining the relative merits of each.
2. Describe the phases of program translation from source code to executable code and the files produced by these phases.
3. Explain the differences between machine-dependent and machine-independent translation.
4. Show the manner in which these differences are evident in the translation process.

CE-SWE8 Software project management [elective]

Topics:
- Team management: team processes; team organization and decision-making, roles and responsibilities in a software team; role identification and assignment; project tracking; team problem resolution
- Project scheduling
- Software measurement and estimation techniques
- Risk analysis
- Software quality assurance
- Software configuration management
- Project management tools

Learning outcomes:
1. Demonstrate through involvement in a team project the central elements of team building and team management.
2. Prepare a project plan for a software project that includes estimates of size and effort, a schedule, resource allocation, configuration control, change management, and project risk identification and management.
3. Compare and contrast the different methods and techniques used to assure the quality of a software product.

CE-SWE9 Software fault tolerance [elective]

Topics:
- Software reliability models
- Software fault-tolerance methods: N-version programming, recovery blocks, rollback and recovery
- Fault tolerance in operating systems and data structures
- Fault tolerance in database and distributed systems

Learning outcomes:
1. Understand the concept of software faults and reliability of software.
2. Understand various redundancy methods used to allow software to detect software faults and produce correct results in the presence of software faults.
3. Understand software fault tolerance approaches used in operating systems, database systems, and distributed systems.

VLSI Design and Fabrication (CE-VLS)

CE-VLS0 History and overview [core]
CE-VLS1 Electronic properties of materials [core]
CE-VLS2 Function of the basic inverter structure [core]
CE-VLS3 Combinational logic structures [core]
CE-VLS4 Sequential logic structures [core]
CE-VLS5 Semiconductor memories and array structures [core]
CE-VLS6 Chip input/output circuits [elective]
CE-VLS7 Processing and layout [elective]
CE-VLS8 Circuit characterization and performance [elective]
CE-VLS9 Alternative circuit structures/low power design [elective]
CE-VLS10 Semi-custom design technologies [elective]
CE-VLS11 ASIC design methodology [elective]

CE-VLS0 History and overview [core]
Minimum core coverage time: 1 hour

Topics:
- Indicate some reasons for studying VLSI and ASIC design
- Highlight some people that influenced or contributed to the area of VLSI and ASIC design
- Indicate some important topic areas such as MOS transistors, inverter structure, circuit performance, combinational and sequential circuits, memory and array structures, chip I/O design, and application-specific integrated circuits
- Describe a transistor and relate it to a semiconductor
- Indicate the characteristics of a MOS transistor
- Describe CMOS transistors and contrast them with MOS technologies
- Describe some sequential logic circuits such as latches and clock distribution
- Describe the structure of memory design
- Contrast memory structures with array structures
- Contrast the advantages of SRAM and DRAM memory devices
- Describe at which point a circuit becomes a chip
- Provide some examples of application-specific integrated circuits
- Explore some additional resources associated with VLSI and ASIC design
- Explain the purpose and role of VLSI and ASIC design in computer engineering

Learning outcomes:
1. Identify some contributors to VLSI and ASIC design and relate their achievements to the knowledge area.
2. Define a semiconductor.
3. Explain the difference between MOS and CMOS transistors.
4. Define a sequential circuit.
5. Identify some memory devices related to VLSI circuits.
6. Define the meaning of a chip.
7. Give an example of an ASIC chip design.
8. Describe how computer engineering uses or benefits from VLSI and ASIC design.

CE-VLS1 Electronic properties of materials [core]
Minimum core coverage time: 2 hours

Topics:
- Solid-state materials
- Electronics and holes
- Doping, acceptors and donors
- p- and n-type material
- Conductivity and resistivity
- Drift and diffusion currents, mobility and diffusion

Learning outcomes:
1. Understand the current carrying mechanism and the I/V characteristics of intrinsic and doped semiconductor materials.
2. Understand the behavior and the I/V characteristics of a reverse-biased and forward-biased PN junction.
3. Understand the function of a PMOS and a NMOS field effect transistor (FET).
4. Indicate a method to model that function using the device equations.
5. Understand the effect of sub-micron device sizes on the function of MOSFETS.
6. Understand the origin and effect of parasitic resistances and capacitances within the transistor itself.

CE-VLS2 Function of the basic inverter structure [core]
Minimum core coverage time: 3 hours

Topics:
- Connectivity, layout, and basic functionality of a CMOS inverter
- The CMOS inverter voltage transfer characteristic (VTC)
- Analysis of the CMOS VTC for switching threshold, V_{OH}, V_{OL}, V_{IH}, V_{IL}, and noise margins
- Effect of changing the inverter configuration on the CMOS VTC
- Connectivity and basic functionality of a Bipolar ECL inverter (optional)
- Connectivity and basic functionality of a Bipolar TTL inverter (optional)

Learning outcomes:
1. Understand the basic functionality of the CMOS inverter.
2. Understand how the VTC of a CMOS inverter is derived from the PMOS and NMOS characteristic I_D vs. V_{DS} family of curves.
3. Analyze the VTC to determine switching threshold, V_{OH}, V_{OL}, V_{IH}, V_{IL}, and Noise Margins.
4. Understand how these quantities reflect the ability of the inverter to operate in the presence of noise.
5. Understand how changing the configuration of the inverter and the MOSFETS that make it up changes the VTC and thus the inverter's operation.
6. Understand the functionality of bipolar-based logic gates. (Optional)

CE-VLS3 Combinational logic structures [core]
Minimum core coverage time: 1 hour

Topics:
- Basic CMOS gate design
- Layout techniques for combinational logic structures
- Transistor sizing for complex CMOS logic devices
- Transmission gates
- Architectural building blocks (multiplexers, decoders, adders, counters, multipliers)

Learning outcomes:
1. Understand the method to perform circuit design for CMOS logic gates.
2. Understand the techniques, such as Euler paths and stick diagrams, used to optimize the layout of CMOS logic circuits.
3. Understand how the size for each transistor in a CMOS logic gate can be determined.
4. Understand the functionality of the CMOS transmission gate.
5. Demonstrate how to use these gates in several logic functions (e.g. multiplexers, transmission gate-based XOR gates).
6. Understand the functionality of several of the more important architectural building blocks identified above.
7. Demonstrate how to optimize these blocks for CMOS implementation.

CE-VLS4 Sequential logic structures [core]
Minimum core coverage time: 1 hour

Topics:
- Storage mechanisms in CMOS logic
- Dynamic latch circuits
- Static latch and flip-flop circuits
- Sequential circuit design
- Single and multiphase clocking
- Clock distribution, clock skew

Learning outcomes:
1. Understand how to use charge storage (capacitance) and feedback to store values in CMOS logic.
2. Understand the circuit design, functionality, advantages, and disadvantages of dynamic latches in CMOS.
3. Understand the circuit design, functionality and advantages and disadvantages of static latches and flip-flops (including edge-triggered) in CMOS.
4. Understand the concepts of bi-stability and metastability in static flip-flops.
5. Understand how latches and flip-flops are used in the design of state machines and data paths
6. Understand the functionality, advantages, and disadvantages of single phase clocking, both level sensitive and edge triggered.
7. Understand the functionality, advantages, and disadvantages of multi (two) phase clocking.
8. Understand the problems arising from clock skew and how one can use clock distribution schemes (including the use of PLLs) to solve it.

CE-VLS5 Semiconductor memories and logic arrays [core]
Minimum core coverage time: 2 hours

Topics:
- Latches
- Flip-flops
- Dynamic read-write memory (DRAM) circuits
- Static read-write memory (SRAM) circuits
- Memory system organization
- Read-only memory circuits
- EPROM/EEPROM/Flash memory circuits
- Programmable Logic Array (PLA) circuits
- FPGA and related devices
- Sense amplifiers

Learning outcomes:
1. Understand how we organize memory systems and why we do not typically organize them in the most simplistic arrangement such as in a one-dimensional word array.
2. Understand the circuit-level implementations possible for read-only memory (ROM) organizations.
3. Understand the layout and function of the specialized transistors used in non-volatile ROM devices and how their characteristics influence the circuit-level implementations of ROMs using them.
4. Understand the functionality and layout of cells used to implement static RAM (SRAM) memories.
5. Understand how SRAMs are typically organized and how their associated peripheral circuitry (sense amps, decoders, address translation detectors, etc.) is organized and functions.
6. Understand how a typical 3-transistor and 1-transistor DRAM cell functions and how to represent them.
7. Understand how DRAMs are typically organized and accessed, and how their associated peripheral circuitry (sense amps, decoders, etc.) is organized and functions.
8. Understand how PLAs function, how can be implemented in CMOS, and how logic functions are mapped to them.

CE-VLS6 Chip input/output circuits [elective]

Topics:
- General I/O pad issues
- Bonding pads
- ESD Protection circuits
- Input, Output, Bidirectional, and analog pads
- VDD and VSS pads

Learning outcomes:
1. Understand the unique functions that I/O circuits must perform and their general circuit-level implementations.
2. Understand the functions of signal I/O pads and their general transistor-level implementations.
3. Understand the functions of VDD and VSS pads for both the core and padframe, and their general transistor-level implementations.

CE-VLS7 Processing and layout [elective]

Topics:
- Processing steps for patterning SiO_2 on a silicon wafer
- CMOS processing technology steps and their results
- Layout design rules and their objectives
- Scalable (-based) design rules
- Design-rule checking

Learning outcomes:
1. Understand the basic steps of photolithography, its limitations, and how that determines minimum line width and device sizes.
2. Understand the processing steps required for fabrication of CMOS devises and the general results of each step.
3. Understand the physical defects that can arise in silicon processing and how design rules attempt to minimize their effects.
4. Understand the spacing and minimum device sizes specified by a typical set of design rules.
5. Understand the benefits and tradeoffs of a -based scalable design rule.
6. Understand the process and tools used for design rule checking.

CE-VLS8 Circuit characterization and performance [elective]

Topics:
- Switching characteristics (rise and fall times, gate delays)
- Power dissipation
- Resistance and capacitance estimation
- CMOS transistor sizing
- Conductor sizing

Learning outcomes:
1. Understand the basic causes of propagation delay and power dissipation in CMOS logic.
2. Understand the techniques for estimating parasitic resistance and capacitance for various layers on a CMOS integrated circuit.
3. Understand the effects of changing (and optimizing) the transistor widths in CMOS logic.
4. Understand the effects of changing (and optimizing) the conductor widths on a CMOS integrated circuit.

CE-VLS9 Alternative circuit structures/low power design [elective]

Topics:
- NMOS
- Pseudo-NMOS,
- Domino-CMOS
- CVSL
- Low power design

Learning outcomes:
1. Understand how to implement MOSFET-based logic families other than CMOS.
2. Understand the advantages and disadvantages of these logic families.
3. Understand the reasons for dynamic and static leakage power.
4. Understand how to design CMOS circuits for low power.

CE-VLS10 Semi-custom design technologies [elective]

Topics:
- Full custom methodology
- Standard cell methodology
- Gate array technologies
- Structured ASICs
- Programmable logic technologies
- Field-programmable gate arrays (FPGAs)
- Time to market and design economics

Learning outcomes:
1. Understand the different design techniques, methodologies, and implementation technologies available to implement a function on a single integrated circuit.
2. Understand the advantages and disadvantages of each technique.
3. Demonstrate how a designer might go about selecting a specific technique for his or her current project.

CE-VLS11 ASIC design methodology [elective]

Topics:

- ASIC design flow (custom, semicustom)
- Design hierarchy
- Computer-aided design (CAD): design modeling and capture (schematic, HDL); design verification (formal, simulation, timing analysis); automated synthesis; layout, floorplanning, place and route; back annotation
- Semi-custom design with programmable logic devices and programmable gate arrays
- System-on-chip (SOC) design and intellectual property (IP) cores
- Testing and design for testability
- Verification

Learning outcomes:

1. Understand the more detailed design issues present in implementing a given digital system on an application-specific integrated circuit (ASIC)
2. Understand the function, capabilities, and disadvantages of the various computer-aided design (CAD) tools available to the ASIC designer to automate portions of the design process.
3. Understand the issues that come with implementing a real world, complex design in an ASIC for a production environment.
4. Understand the basic principles of test generation and design for testability
5. Understand the difference between testing and verification

Appendix B

Computer Engineering
Sample Curricula

This appendix to the *Computing Curricula - Computer Engineering (CE2004)* report contains four example curricula that illustrate possible implementations of four-year degree programs that satisfy the required specifications of the body of knowledge detailed in the main body of the report. These implementations illustrate how undergraduate programs of different flavors and of different characteristics may be effectively implemented to suit different institutional requirements and resource constraints, and hence serve a wide variety of educational goals and student needs. None of these examples is intended to be prescriptive.

The table below summarizes the sample curricula. This table can serve as a guide to identifying sample curricula that are most relevant to particular institutional needs and priorities.

Implementation	Administrative Entity
A	Computer Science Department
B	Electrical & Computer Engineering Department
C	Joint - Computer Science and Electrical Engineering Departments
D	United Kingdom

B.1 Format and Conventions

All four sample curricula presented in this appendix are presented using a common format, with five logical components:

1. A set of educational objectives for the program of study and an explanation of any institutional, college, department, or resource constraints that are assumed;
2. A summary of degree requirements, in tabular form, to indicate the curricular content in its entirety;
3. A sample four-year schedule that a typical student might follow;
4. A map showing coverage of the Computer Engineering Body of Knowledge by courses in the curriculum;
5. A set of course descriptions for those courses in the computing component of the curriculum.

To clarify the identification of courses, levels, and implementations, each course is numbered in a way that identifies the implementation in which it appears and the level at which it is normally taught. Thus, a course numbered CSC_X100 is a course in implementation X that is commonly taught in the first year (at the freshman level); CSC_X200 is a course that is commonly taught in the second year (at the sophomore level); CSC_X300 is a course that is commonly taught in the third year (at the junior level); and course CSC_X400 is commonly taught in the fourth year (at the senior level).

To provide ease of comparison, all four implementations are presented as a set of courses, designed for a U.S. semester system in which a semester provides 14 weeks of lecture, lab, and recitation (in-class problem solving) time. This does not typically include time for final examination, vacations, and reading periods. For simplicity, lecture, lab, and recitation times are specified in “hours”, where one “hour” of lecture, lab, or recitation is typically 50-minutes in duration.

Each course is assigned a number of semester credit hours, according to the number and types of formal activities within a given week. These are determined as follows.

- Lecture hours: presentation of material in a classroom setting
  - 1 credit hour = 1 "hour" of lecture per week
- Laboratory hours: formal experimentation in a laboratory setting
  - 1 credit hour = one 3-"hour" laboratory session per week
- Recitation hours: problem-solving sessions, programming, etc. in support of lecture material
  - 1 "hour" of recitation per week is assigned no additional credit

Examples:

3-credit lecture course
3 lecture "hours" per week for 14 weeks = 42 lecture "hours"

3-credit lecture course with a 1-hour recitation session per week
3 lecture "hours" per week for 14 weeks = 42 lecture "hours"
1 recitation "hour" per week for 14 weeks = 14 recitation "hours"

1-credit laboratory course:
One 3-"hour" laboratory session per week for 14 weeks = 42 lab "hours"

3-credit course with two lectures and a lab session each week:
2 lecture "hours" per week for 14 weeks = 28 lecture "hours"
One 3-"hour" lab per week for 14 weeks = 42 lab "hours"

3-credit senior project design course
1 classroom meeting per week for 14 weeks = 14 lecture "hours"
2 credits of "laboratory" = 6 "hours" of laboratory per week for 14 weeks = 84 lab "hours"

B.2 Preparation to Enter the Profession

The four sample curricula in this appendix have as a major goal the preparation of graduates for entry into the computer engineering profession. There are many ways of building a curriculum whose graduates are well-educated computer engineers. To emphasize this point, the four programs of study outlined in this section are quite distinct. These programs differ in their emphasis and in the institutional constraints.

These curricula are designed to ensure appropriate coverage of the core topics of the computer engineering BOK as defined in this report. However, as also discussed in the main report, there are many other elements to creating a program that will effectively prepare graduates for the professional practice of computer engineering, such as design and laboratory experience, oral and written communication, and usage of modern engineering tools. Accordingly, professional accreditation addresses more than just curriculum, and readers interested in accreditation should consult the relevant criteria from the local accrediting agency (EAC/ABET [1], etc.) for complete accreditation criteria.

In addition, each individual computer engineering program may have educational objectives that are unique to that program and not directly reflected in the computer engineering BOK and curriculum models presented in this report. It is the responsibility of each program to ensure that its students achieve each learning outcome essential to the educational objectives of the program.

B.3 Curricula Commonalities

Students desiring to study the application of computers and digital systems will find computer engineering to be a rewarding experience. Study is intensive and students desiring to develop proficiency in the subfields of computer engineering such as hardware, software, and systems that arise in the design,

analysis, development, and application of computers and digital systems, will find this program to be a challenge. Applied skills will enable students to analyze, design, and test digital computer systems, architectures, networks, and processes.

Each sample curriculum leads to a bachelor's degree in computer engineering and provides a balanced treatment of hardware and software principles; each provides a broad foundation in some combination of computer science and electrical/electronic engineering of computers and digital systems with emphasis on theory, analysis, and design. Additionally, each of the first three curricula samples (Curricula A, B, and C) provides a broad foundation in the sciences, discrete and continuous mathematics, and other aspects of a general education. The last sample curriculum model (Curriculum D) demonstrates a typical program in computer engineering as one might find in the United Kingdom, Europe, Asia, and other parts of the world. The demonstrated three-year degree program would lead to a typical Bachelor of Engineering degree. The four-year curriculum would lead to a typical Master's of Engineering degree.

The common requirements spread widely across a range of courses and allow revisiting the subject matter with spiral learning taking place. Each curriculum contains sufficient flexibility to support various areas of specialization. Each program structure allows a broadly based course of study and provides selection from among many professional electives. In all cases, the capstone design experience takes place after students in the program have developed sufficient depth of coverage in the core subject areas. A combination of theory, practice, application, and attitudes accompany the construction of each course.

The goals of each program are to prepare students for a professional career in computer engineering by establishing a foundation for lifelong learning and development. It also provides a platform for further work leading to graduate studies in computer engineering as well as careers in fields like business, law, medicine, management and others. Students develop design skills progressively, beginning with their first courses in programming, circuit analysis, digital circuits, computer architectures, and networks and they apply their accumulating knowledge to practical problems throughout the curriculum. The process culminates in the capstone design course, which complements the analytical part of the curriculum. The thorough preparation afforded by the computer engineering curriculum includes the broad education necessary to understand the impact of engineering solutions in a global and societal context.

Graduates of each program should be well prepared for professional employment or advanced studies. They should understand the various areas of computer engineering such as applied electronics, digital devices and systems, software design, and computer architectures, systems, and networks. Graduates should be able to apply their acquired knowledge and skills to these and other areas of computer engineering. They will also possess design skills and have a deep understanding of hardware issues, software issues, models, the interactions between these issues, and related applications. The thorough preparation afforded by this computer engineering curriculum includes the broad education necessary to understand the impact of engineering solutions in a global and societal context.

B.4 Curriculum Implementation A

A Computer Engineering Program Administered by a Typical Computer Science Department

B.4.1 Program Goals and Features

A computer science department would sponsor this B.S. program in computer engineering. This program assumes an evolution from a computer science program, and therefore it might be of interest to schools that have a computer science department but not a large engineering program. As is typical of many computer science programs, this model has a larger general education component than the other curricula presented in this appendix, and therefore fewer hours devoted to computer engineering topics. Consequently, several courses have been designed specifically to provide coverage of the core topics of the CPE BOK. In particular, courses that cover the traditional EE topics in the CPE BOK core have been designed to cover the core material without going significantly beyond these topics. In contrast, a number of the computer science courses do go beyond the core material, although not as much as in the other three curricula presented in this appendix.

B.4.2 Summary of Requirements

This program of study builds around a set of eleven required courses in computer science (including a culminating design project) and four from electrical engineering, comprising 47 credit hours of study. The program achieves flexibility through a judicious choice of three technical electives and a capstone project. The computer engineering segment of the curriculum, including technical electives and capstone, comprises 56 credit-hours of study. Laboratory experience is provided in the first two introductory computer science courses and in the circuits and digital logic courses. The total number of hours devoted to laboratory experience is less than in the other three curriculum models presented in this appendix. In addition, since there are fewer courses that incorporate engineering design, the capstone design project course extends two full semesters (six credits) to ensure that all graduates have significant design experience, as well as experience with teamwork and modern engineering tools. Oral and written communication skills are assumed to be emphasized in both the general education and in the computer engineering segments of this curriculum.

This curriculum utilizes a relatively traditional course structure and content. It requires 39 courses, with credit hours distributed as follows:

Credit-hours	Topic areas
20	Mathematics and statistics
12	Basic science (physics, chemistry)
33	Humanities, social sciences, composition, and literature
27	Required computer science
15	Required electrical engineering
5	Design project (from computer science)
9	Technical electives (from computer science or engineering)
3	Free electives
124	*TOTAL Credit Hours for Computer Engineering Program*

B.4.3 Four-Year Curriculum Model for Curriculum A

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 1			**Semester 2**		
MTH 101	Calculus I	4	MTH 102	Calculus II	4
PHY 101	Physics I	4	PHY 102	Physics II	4
CSC_A101	Computer Science I	4	CSC_A102	Computer Science II	4
	English Composition	3		Humanities / Social Science	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 3			**Semester 4**		
MTH 201	Differential Equations	3	MTH 202	Linear Algebra	3
CHM 201	Chemistry I	4	CSC_A202	Computer Organization	3
CSC_A201	Algorithm Design	3	ELE_A202	Circuits and Systems	4
ELE_A201	Digital Logic	4		Humanities / Social Science	3
	Humanities / Social Science	3		Humanities / Social Science	3
	Total Credit Hours	*17*		*Total Credit Hours*	*16*
Semester 5			**Semester 6**		
MTH_A301	Discrete Structures	3	MTH_A302	Probability and Statistics	3
CSC_A301	Computer Architecture	3	CSC_A302	Embedded Systems I	3
ELE_A301	Electronics	4	CSC_A303	Computer Networks	3
	Technical Writing	3	ELE_A302	Digital Signal Processing	3
	Humanities / Social Science	3		Humanities / Social Science	3
	Total Credit Hours	*16*		*Total Credit Hours*	*15*
Semester 7			**Semester 8**		
CSC_A401	Embedded Systems II	3	CSC_A404	Senior Project II	3
CSC_A402	Computer Ethics	1		Technical elective	3
CSC_A403	Senior Project I	2		Technical elective	3
	Technical elective	3		Humanities / Social Science	3
	Humanities / Social Science	3		Free Elective	3
	Humanities / Social Science	3			
	Total Credit Hours	*15*		*Total Credit Hours*	*15*

B.4.4 Mapping of Computer Engineering BOK to Curriculum A

BOK Area / Course	ALG	CAO	CSE	CSG	DBS	DIG	DSC	DSP	ELE	ESY	HCI	NWK	OPS	PRF	PRS	SPR	SWE	VLS
CSC_A101					0-2									0-4				
CSC_A102											0-2			5-8			0-2	
CSC_A201	0-5										3-4						3-6	
CSC_A202		0-4				0-1												
CSC_A301		5-9																
CSC_A302			0-8										0-4					
CSC_A303												0-8						
CSC_A401						6-7				0-7								
CSC_A402																0-8		
CSC_A403																		
CSC_A404																		
ELE_A201						0-6 8,9												
ELE_A202				0-7														
ELE_A301									0-10									0-6
ELE_A302				8				0-11										
MTH_A301							0-6											
MTH_A302															0-7			
Required BOK Hours	30	63	18	43	5	57	33	17	40	20	8	21	20	39	33	16	13	10
Required BOK Units	0-5	0-9	0-8	0-6	0-2	0-9	0-6	0-6	0-10	0-6	0-4	0-6	0-4	0-5	0-7	0-8	0-6	0-5
Elective BOK Units	6	10	9-11	7-9	3-8	10		7-11	11-14	7-10	5-10	7-10	5-8	6-8	8	9	7-9	6-11

B.4.5 Curriculum A – Course Summaries

MTH_A301: Discrete Structures

Review of propositional and predicate logic; methods of theorem proving; strong and weak induction; finite and infinite sets, set operations; introductions to computational complexity, theta and big-O notation; combinatorics, including permutations and combinations; discrete probability and binomial distribution.

Prerequisites: Pre-calculus or equivalent.
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14
CE2004 BOK Coverage: CE-DSC 0-6.

CSC_A101: Computer Science I

Introduction to computing; algorithmic thinking, data structures, data transformation and processing, and programming in a media and communication context.

Prerequisites: Pre-calculus or equivalent.
Credit Hours: 4 Lecture Hours: 42 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-DBS 0-2, CE-PRF 0-4.

CSC_A102: Computer Science II

Second course in programming languages and systems. Topics include data structures, assemblers, compilers, and syntactical methods; recursion, string manipulation and list processing; concepts of executive programs and operating systems; introduction to time-sharing systems.

Prerequisites: CSC_A101
Credit Hours: 4 Lecture Hours: 42 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-HCI 0-2, CE-PRF 5-8, CE-SWE 0-2.

CSC_A201: Algorithm Design

Design and analysis techniques for solving domain specific problems, algorithm design strategies, distributed algorithms.

Prerequisites: CSC_A102
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-ALG 0-5, CE-HCI 3-4, CE-SWE 3-6.

CSC_A202: Computer Organization

Introductory course in computer organization and architecture. Topics include basic hardware and software structure, addressing methods, programs control, processing units, I-O organization, arithmetic, main-memory organization, peripherals, microprocessor families, RISC architectures, and multiprocessors.

Prerequisites: CSC_A101
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-CAO 0-4, CE-DIG 0-1.

CSC_A301: Computer Architecture

Topics include a review of logic circuits, bus structures, memory organization, interrupt structures, arithmetic units, input-output structures, state generation, central processor organization, control function implementation, and data communication, design of digital systems.

Prerequisites: CSC_A202
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-CAO 5-9.

CSC_A302: Embedded Systems I

Interfacing of microcomputers to peripherals or other computers for purposes of data acquisition, device monitoring and control, and other communications. The interfacing problem is considered at all levels including computer architecture, logic, timing, loading, protocols, and software laboratory for building and simulating designs.

Prerequisites: CSC_A202
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14
CE2004 BOK Coverage: CE-CSE 0-8, CE-OPS 0-4.

CSC_A303: Computer Networks

Introduction to the design and performance analysis of local computer networks. Emphasis is on performance analysis of representative multi-access procedures.

Prerequisites: CSC_A102, MTH 102
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14
CE2004 BOK Coverage: CE-NWK 0-8.

CSC_A401: Embedded Systems II

Design of embedded digital systems; microcontrollers, embedded programs, real-time operating systems, design methodologies, hardware-software codesign, hardware modeling and computer-aided design, prototyping with FPGAs.

Prerequisites: CSC_A302
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14
CE2004 BOK Coverage: CE-ESY 0-7, CE-DIG 6-7

CSC_A402: Computer Ethics

Critical examination of ethical problems associated with computer science and engineering. Legal and quasi-legal (i.e., policy and regulative) issues are also considered. Topics addressed include the process of ethical decision-making, privacy and confidentiality, computer crime, professional codes and responsibilities, software piracy, the impact of computers on society.

Prerequisites: Junior standing
Credit Hours: 1 Lecture Hours: 14 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-SPR 0-8

CSC_A403: Senior Project I

Individually designed projects oriented toward providing experience in the establishment of objectives, criteria, synthesis, analysis, construction, testing, and evaluation; solution of open-ended problems; design methodology.

Prerequisites: CSC_A301, CSC_A302
Credit Hours: 2 Lecture Hours: 14 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: None.

CSC_A404: Senior Project II

Continuation of Senior Project I focused upon implementation of a project design.

Prerequisites: CSC_B403
Credit Hours: 3 Lecture Hours: 14 Lab Hours: 84 Recitation Hours: 0
CE2004 BOK Coverage: None.

ELE_A201: Digital Logic

Study of logic with an introduction to Boolean algebra; number systems and representation of information; use of integrated circuits to implement combinational and sequential logic functions and computing elements; organization and structure of computing systems.

Prerequisites: CSC_A101
Credit Hours: 4 Lecture Hours: 42 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK *Coverage:* CE-DIG 0-6, 8-9.

ELE_A202: Circuits and Systems

DC resistive circuits, Kirchhoff's Laws, Nodal and Mesh emphasis, sources, Thevenin's and Norton's theorems, RC, RL, RCL circuit solutions with initial condition using homogenous or nonhomogenous ordinary differential equations having constant coefficients, sinusoidal steady state solution, three-phase circuits, complex frequency and network functions, frequency response, two-port parameters, magnetically-coupled circuits, Laplace transforms, and introduction to Fourier series and transforms..

Prerequisites: MTH 201, PHY 102
Credit Hours: 4 Lecture Hours: 42 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-CSG 0-7

ELE_A301: Electronics

Introduction to electronic materials and devices; principles of design; design of DC and AC circuits using diodes, bipolar junction transistors, field-effect transistors and use of transistors in digital circuits, physical design of simple gates, flip-flops, and memory circuits.

Prerequisites: ELE_A202
Credit Hours: 4 Lecture Hours: 56 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-ELE 0-10, CE-VLS 0-6

ELE_A302: Digital Signal Processing

Digital processing of signals, sampling, difference equations, discrete-time Fourier transforms, discrete and fast Fourier transforms, digital filter design.

Prerequisites: ELE_A202
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-DSP 0-11, CE-CSG 8

B.5 Curriculum Implementation B

A Computer Engineering Program Administered by an Electrical and Computer Engineering Department

B.5.1 Program Goals and Features

This program leads to a bachelors degree in computer engineering, as might be offered by a traditional electrical and computer engineering (ECE) department. Foundation courses in computer science typically are offered by a computer science department; the remaining courses taught by the ECE department. As is typical of most programs in engineering, this program has a smaller general education component than Curriculum A described in this appendix, with more hours devoted to computer engineering topics. This program is characterized by a greater emphasis on some of the traditional electrical engineering topics (circuits, electronics) than the other curricula in this appendix, providing coverage well beyond the minimum recommended core coverage of these topics. However, coverage of such topics as computer architecture, embedded systems, software design, and related computer science topics is also significant. Graduates should be well prepared to pursue careers that entail hardware and system design (VLSI and ASIC design, embedded systems, networks, etc.), with sufficient background in software to enable them to be effective computer engineers.

B.5.2 Summary of Requirements

This program of study builds around a set of eighteen required courses in computer engineering (and electrical engineering), comprising 49 credit hours of study. It incorporates flexibility through choice of nine credits of electives in electrical and computer engineering. Design and the use of modern tools are emphasized throughout the curriculum, culminating in a one-semester capstone design course. The laboratory experience includes four required lab courses in electric circuits, digital logic circuits, computer systems, and digital systems design, and provides additional opportunities for team-based projects. Oral and written communication skills are addressed in required oral communication and technical writing courses, and are assumed to be reinforced throughout the laboratory courses and the capstone project.

The computer engineering segment of the curriculum, including professional electives, comprises 58 semester credit hours of study, supported by 9 additional hours of required computer science courses. This curriculum requires 43 courses, with credit hours distributed as follows:

Credit-hours	Topics
18	Mathematics
12	Basic Science (Physics, Chemistry)
21	English composition, humanities and social sciences
9	Required computer science
49	Required electrical and computer engineering
9	Elective electrical and computer engineering
6	Other engineering courses
124	*TOTAL Credit Hours for Computer Engineering Program*

B.5.3 Four-Year Curriculum Model for Curriculum B

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 1			**Semester 2**		
MTH 101	Calculus I	3	MTH 102	Calculus II	3
CHM 101	Chemistry I	3	CHM 102	Chemistry II	3
CSC_B101	Programming & Prob. Solving I	3	CSC_B102	Programming & Prob. Solving II	3
	English Composition I	3	ENG 101	Engineering Problem Solving	3
	Humanities & Social Science	3		English Composition II	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 3			**Semester 4**		
MTH 201	Calculus III	3	MTH 203	Linear Differential Equations	3
MTH_B202	Discrete Structures	3	CSC_B201	Algorithms & Data Structures	3
PHY 201	Physics I	3	ECE_B204	Signals & Systems	3
ECE_B201	Digital Logic Circuits	3	ECE_B205	Electric Circuits Lab	1
ECE_B202	Electric Circuits	3	ECE_B206	Digital Electronics	3
ECE_B203	Digital Logic Lab	1	ECE_B207	Computer Organization	3
	Total Credit Hours	*16*		*Total Credit Hours*	*16*
Semester 5			**Semester 6**		
MTH 301	Linear Algebra	3	ECE_B304	Digital System Design	3
PHY 301	Physics II	3	ECE_B305	Digital Systems Lab	1
ECE_B301	Operating Systems	3	ECE_B306	Analog Electronics	3
ECE_B302	Computer Architecture	3	ECE_B307	Random Signals & Systems	3
ECE_B303	Computer Design Lab	1		Fine Arts Elective	3
	Humanities & Soc Science II	3		Oral Communication	3
	Total Credit Hours	*16*		*Total Credit Hours*	*16*
Semester 7			**Semester 8**		
ECE_B401	Fundamentals of Software Design	3	ECE_B403	Computer Networks	3
ECE_B402	Embedded Computing Systems	3	ECE_B404	Ethics, Society, Profession	3
ENG 401	Engineering Economics	3	ECE_B405	Senior Design Project	3
	ECE Elective I	3		ECE Elective II	3
	Technical Writing	3		ECE Elective III	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*

B.5.4 Mapping of Computer Engineering BOK to Curriculum B

BOK Area / Course	ALG	CAO	CSE	CSG	DBS	DIG	DSC	DSP	ELE	ESY	HCI	NWK	OPS	PRF	PRS	SPR	SWE	VLS
CSC_B101														0-4				
CSC_B102														5-8				
CSC_B201	0-6				0-2													
ECE_B201						0-5												
ECE_B202				0-5														
ECE_B203						0-5												
ECE_B204				6-9				0-6										
ECE_B205				0-3														
ECE_B206									0-7									0-6
ECE_B207		0-3, 9				6-7												
ECE_B301		3,5,8											0-9					
ECE_B302		3-10				6												
ECE_B303		3-5																
ECE_B304			0-7			7												10,11
ECE_B305			0-7			7												10,11
ECE_B306									9-14									
ECE_B307															0-7			
ECE_B401			0-8								0-4						0-7	
ECE_B402		5							8	0-10								
ECE_B403												0-8						
ECE_B404																0-8		
ECE_B405																		
MTH_B202							0-6											
Required BOK Hours	30	63	18	43	5	57	33	17	40	20	8	21	20	39	33	16	13	10
Required BOK Units	0-5	0-9	0-8	0-6	0-2	0-9	0-6	0-6	0-10	0-6	0-4	0-6	0-4	0-5	0-7	0-8	0-6	0-5
Elective BOK Units	6	10	9-11	7-9	3-8	10		7-11	11-14	7-10	5-10	7-10	5-8	6-8	8	9	7-9	6-11

B.5.5 Curriculum B – Course Summaries

MTH_B202: Discrete Structures

Review of propositional and predicate logic; methods of theorem proving; strong and weak induction; finite and infinite sets, set operations; introductions to computational complexity, theta and big-O notation; combinatorics, including permutations and combinations; discrete probability and binomial distribution.

Prerequisites: Pre-calculus or equivalent.

Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14

CE2004 BOK Coverage: CE-DSC 0-6.

CSC_B101: Programming and Problem Solving I

First course in programming languages, syntax, fundamental data structures, algorithms and basic problem-solving.

Prerequisites: Pre-calculus or equivalent

Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14

CE2004 BOK Coverage: CE-PRF 0 – 4

CSC_B102: Programming and Problem Solving II

Second course in programming languages and systems. Topics include assemblers, compilers, and syntactical methods; string manipulation and list processing; concepts of executive programs and operating systems; introduction to time-sharing systems.

Prerequisites: CSC_B101

Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14

CE2004 BOK *Coverage:* CE-PRF 5 – 8

CSC_B201: Algorithms and Data Structures

Design and analysis techniques for solving domain specific problems, algorithm design strategies, distributed algorithms, introduction to database systems and data modeling.

Prerequisites: CSC_B102, MTH 202

Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0

CE2004 BOK *Coverage:* CE-ALG 0 – 6, CE-DBS 0-2

ECE_B201: Digital Logic Circuits

Study of logic with an introduction to Boolean algebra; number systems and representation of information; use of integrated circuits to implement combinational and sequential logic functions and computing elements; organization and structure of computing systems.

Prerequisites: CSC_B101

Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0

CE2004 BOK *Coverage:* CE-DIG 0 – 5

ECE_B202: Electric Circuits

DC resistive circuits, Kirchhoff's Laws, Nodal and Mesh emphasis, sources, Thevenin's and Norton's theorems, RC, RL, RCL circuit solutions with initial condition using homogenous or nonhomogenous ordinary differential equations having constant coefficients. Develop sinusoidal steady state solution.

Prerequisites: MTH 102

Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0

CE2004 BOK Coverage: CE-CSG 0 – 5

ECE_B203: Digital Logic Laboratory

Hands-on experience in using digital electronics by way of logic gates and integrated circuits; practical construction, testing, and implementation of combinational and sequential logic circuits.

Corequisite: ECE_B201

Credit Hours 1 Lecture Hours: 0 Lab Hours: 42 Recitation Hours: 0

CE2004 BOK Coverage: CE-DIG 1-5

ECE_B204: Signals & Systems

Time-domain and frequency-domain methods for modeling and analyzing continuous and discrete-data signals and systems, Laplace transforms, Fourier series and transforms, sampling, discrete signals, z-transforms, digital filters.

Prerequisites: ECE_B202
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-CSG 6 – 9, CE-DSP 0-6

ECE_B205: Electrical Circuits Laboratory

Principles of measurement and instruments used to measure parameters and dynamic variables in electric circuits, steady state and transient measurements in DC and AC circuits, and data analysis methods.

Prerequisite: ECE_B202
Credit Hours 1 Lecture Hours: 0 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-CSG 0-3

ECE_B206: Digital Electronics

Introduction to electronic materials and devices; principles of design; design of DC and AC circuits using diodes, bipolar junction transistors, field-effect transistors and use of transistors in digital circuits, including combinational and sequential circuits.

Prerequisites: ECE_B201, ECE_B204
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-ELE 0 –7, CE-VLS 0-6

ECE_B207: Computer Organization

Introductory course in computer organization and architecture. Topics include basic hardware and software structure, addressing methods, programs control, processing units, I-O organization, arithmetic, main-memory organization, peripherals, microprocessor families, RISC architectures, and multiprocessors.

Prerequisites: CSC_B101, ECE_B201
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-CAO 0 – 3, 9, CE-DIG 6-7

ECE_B301: Operating Systems

Basic operating system components and their functions, concurrency, scheduling and dispatch, memory and device management, file systems, and performance evaluation.

Prerequisites: CSC_B201, ECE_B207
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-OPS 0 –9, CE-CAO 3, 5, 8

ECE_B302: Computer Architecture

Computer bus structures, memory organization, interrupt structures, arithmetic units, input-output structures, central processor organization, control function implementation, pipelining, performance measurement, and distributed system models.

Prerequisites: ECE_B207
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-CAO 3–10, CE-DIG 6

ECE_B303: Computer Design Laboratory

Laboratory experiments include interfacing memory and peripheral devices to a microcomputer, the design of software to control these devices, and the integration of computer hardware and software to control a system.

Prerequisites: ECE_B203, ECE_B207
Credit Hours 1 Lecture Hours: 0 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-CAO 3-5

ECE_B304: Digital System Design

Hierarchical modular design of digital systems, design modeling with a hardware description language, functional and timing simulation of digital systems, implementation in programmable logic devices and field-programmable gate arrays, formal verification, fault models and testing. Designs are developed, simulated and implemented in field-programmable gate arrays in laboratory sessions.

Prerequisites: ECE_B207
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-DIG 7, CE-CSE 0–7, VLS 10-11

ECE_B305: Digital Systems Laboratory

Digital system designs are developed, modeled, simulated and implemented in field-programmable gate arrays.

Corequisites: ECE_B304
Credit Hours 1 Lecture Hours: 0 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-DIG 7, CE-CSE 0–7, VLS 10-11

ECE_B306: Analog Electronics

Design and analysis of single-stage and multistage transistor amplifiers; biasing for integrated circuit design; small-signal modeling; operational amplifier circuits; IC design techniques; noise and RF amplifiers; D/A and A/D converters.

Prerequisites: ECE_B206
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-ELE 9-14

ECE_B307: Random Signals & Systems

Introduction to engineering problems of a probabilistic nature. Systems transformations, statistical averages, simulation, and estimation of system parameters.

Prerequisites: ECE_B204
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-PRS 0-7

ECE_B401: Fundamentals of Software Design

Foundations of software design, reasoning about software, the calculus of programs, survey of formal specification techniques and design languages, human-computer interaction, input/output, graphical user interfaces.

Prerequisites: CSC_B201
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-SWE 0 – 7, CE-CSE 0-8, CE-HCI 0-4

ECE_B402: Embedded Computing Systems

Interfacing of microcomputers to peripherals or other computers for purposes of data acquisition, device monitoring and control, and other communications. The interfacing problem is considered at all levels including computer architecture, logic, timing, loading, protocols, and software laboratory for building and simulating designs.

Prerequisites: ECE_B302, ECE_B303
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-ESY 0–10, CE-CAO 5, CE-ELE 8

ECE_B403: Computer Networks

Introduction to the design and performance analysis of computer networks. Architectures, protocols, standards and technologies of computer networks; design and implementation of networks based on requirements; applications of information networks for data, audio and video communications; performance analysis.

Prerequisites: ECE_B207, ECE_B301
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-NWK 0–8

ECE_B404: Ethics, Society, Profession

Critical examination of ethical problems associated with computer engineering. Discussion of these problems is conducted within the framework of classical philosophical ethical theories. Legal and quasi-legal (i.e., policy and regulative) issues are also considered. Topics addressed include the process of ethical decision-making, privacy and confidentiality, computer crime, professional codes and responsibilities, software piracy, the impact of computers on society.

Prerequisites: Junior standing
Credit Hours 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-SPR 0-8

ECE_B405: Design Projects in Computer Engineering

Individually defined projects oriented toward providing experience in establishment of objectives and criteria, synthesis, analysis, construction, testing, and evaluation; development of student creativity through the solution of open-ended problems; individual instruction in design methodology.

Prerequisites: ECE_B304, ECE_B401
Credit Hours 3 Lecture Hours: 14 Lab Hours: 84 Recitation Hours: 0
CE2004 BOK Coverage: none

Sample ECE Elective Courses

ECE_B501: Communications Systems

Study of communication systems design and analysis. Topics include signals and spectra, baseband signaling and detection in noise, digital and analog modulation and demodulation techniques, and communications link budget analysis.

ECE_B502: Digital Electronics

Electronic devices and circuits of importance to digital computer operation and to other areas of electrical engineering are considered. Active and passive waveshaping, waveform generation, memory elements, switching, and logic circuits are some of the topics. Experimentation with various types of circuits is provided by laboratory projects.

ECE_B503: Instrumentation

Theory and analysis of transducers and related circuits and instrumentation. Generalized configurations and performance characteristics of instruments are considered. Transducer devices for measuring physical parameters such as motion, force, torque, pressure, flow, and temperature are discussed.

ECE_B504: Integrated Circuit Design

Design concepts and factors influencing the choice of technology; fundamental MOS device design; silicon foundries, custom and semi-custom integrated circuits; computer-aided design software/hardware trends and future developments; hands-on use of CAD tools to design standard library cells; systems design considerations, testing, and packaging.

ECE_B505: Computer-Aided Analysis and Design

Principles and methods suited to the solution of engineering problems on the digital computer. Topics include widely used methods for the solution of the systems of algebraic and/or differential equations which arise in modeling of engineering systems, data approximation and curve fitting, continuous system simulation languages, and design-oriented programming systems.

ECE_B506: Introduction to Digital Signal Processing

Introduction to characteristics, design, and applications of discrete time systems; design of digital filters; introduction to the Fast Fourier Transform (FFT); LSI hardware for signal processing applications.

ECE_B507: Knowledge Engineering

Introduction to the theoretical and practical aspects of knowledge engineering or applied artificial intelligence. Topics include symbolic representation structures and manipulation, unification, production systems and structures, rule-based and expert systems, planning and AI system architectures; system design in PROLOG and LISP. Project is required.

ECE_B508: Digital Communications

Study of digital communication systems. Topics include error-control coding, synchronization, multiple-access techniques, spread spectrum signaling, and fading channels.

B.6 Curriculum Implementation C

Computer Engineering Program Administered Jointly by a Computer Science Department and an Electrical Engineering Department

B.6.1 Program Goals and Features

A computer science department and an electrical engineering department (or perhaps a general engineering department) jointly sponsor this program leading to a bachelor's degree in computer engineering. This curriculum provides a broad foundation in the science and engineering of computers and digital systems with emphasis on theory, analysis, design, natural science, and discrete and continuous mathematics. The curriculum is characterized by a general education component comparable to that of Curriculum B described earlier, with a balanced coverage of traditional electrical engineering topics, fundamentals of computer science, computer architecture, embedded systems, networks, and software design, as well as the interactions between these elements. The minimum recommended core coverage is exceeded in most of these areas, with four technical electives available to allow students to emphasize a selected area if they choose. Graduates should thus be well prepared to pursue careers that entail the design of hardware, software, and/or systems.

The thorough preparation afforded by the computer engineering curriculum includes the broad education necessary to understand the impact of engineering solutions in a global and societal context. Hence, graduates will be well prepared for professional employment or advanced studies.

B.6.2 Summary of Requirements

This program of study is built around a set of thirteen required courses in computer science and thirteen required courses in electrical engineering, comprising 67 required credit hours of study. The program achieves flexibility through a judicious choice of four technical electives. Design and the use of modern tools are emphasized throughout the curriculum, culminating in a one-semester capstone design course. The laboratory experience includes four required lab courses in electric circuit analysis, digital logic circuits, electronics, and computer architecture, and provides additional opportunities for team-based projects. Oral and written communication skills are addressed in the laboratory courses and the capstone project, as well as in courses on ethics, technology and society, and others.

The computer engineering segment of the curriculum, including professional electives, comprises 79 semester credit-hours of study. This curriculum utilizes a relatively traditional course structure and content. It requires 44 courses, with credit hours distributed as follows:

Credit-hours	Topic areas
15	Mathematics
14	Basic science (physics, chemistry)
21	Humanities, social sciences, composition, literature
35	Required computer science
32	Required electronic engineering
12	Technical electives (computer science or engineering)
129	*TOTAL Credit Hours for Computer Engineering Program*

B.6.3 Four-Year Curriculum Model for Curriculum C

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 1			**Semester 2**		
MTH 101	Calculus I	4	MTH 102	Calculus II	4
CSC_C101	Discrete Structures for Computer Sci.	3	PHY 101	Physics I + Lab	5
CSC_C102	Programming I	3	CSC_C103	Programming II	3
ENG_C101	Introduction to Engineering	2		English Composition II	3
	English Composition I	3		Humanities / Social Science	3
	Total Credit Hours	*15*		*Total Credit Hours*	*18*
Semester 3			**Semester 4**		
MTH 201	Calculus III	4	CHM 201	Chemistry I + Lab	4
PHY 201	Physics II + Lab	5	CSC_C202	Intro to Computer Organization	3
CSC_C201	Algorithms and Data Structures	3	ENG_C203	Circuit Analysis Lab	1
ENG_C201	Engineering Circuit Analysis	3		Literature	3
ENG_C202	Digital and Logic Design	3		Humanities / Social Science	3
				Humanities / Social Science	3
	Total Credit Hours	*18*		*Total Credit Hours*	*17*
Semester 5			**Semester 6**		
MTH 301	Engineering Mathematics I	3	CSC_C302	Computing, Ethics & Society	1
CSC_C301	Operating Systems	3	CSC_C303	Computer Network Communications	3
ENG_C301	Electronic Circuits	3	CSC_C304	Applied Probability and Statistics	3
ENG_C302	Digital and Logic Design Lab	1	ENG_C305	VLSI Design	3
ENG_C303	Embedded Microprocessor Systems	3	ENG_C306	Electronics Lab	1
ENG_C304	Signals and Linear Systems	3		Technical elective	3
				Literature	3
	Total Credit Hours	*16*		*Total Credit Hours*	*17*
Semester 7			**Semester 8**		
CSC_C401	Computer Architecture	3	CSC_C403	Computer Architecture Lab	1
CSC_C402	Software Engineering	3	CSC_C404	Simulation and Modeling	3
ENG_C401	Digital Communication and Signal Processing	3	ENG_C403	Computer Engineering Design	3
ENG_C402	Technology & Society	3		Technical elective	3
	Technical elective	3		Technical elective	3
	Total Credit Hours	*15*		*Total Credit Hours*	*13*

B.6.4 Mapping of Computer Engineering BOK to Curriculum C

BOK Area / Course	ALG	CAO	CSE	CSG	DBS	DIG	DSC	DSP	ELE	ESY	HCI	NWK	OPS	PRF	PRS	SPR	SWE	VLS
CSC_C101							0-6											
CSC_C102														0-4			0-1	
CSC_C103														5-8			2-3	
CSC_C201	0-6																4-5	
CSC_C202		0-4																
CSC_C301					0-3								0-8					
CSC_C302											0-1					0-9		
CSC_C303												0-8						
CSC_C304															0-8			
CSC_C401		3-10																
CSC_C402					0-2						0-4						0-9	
CSC_C403		3-5 8-9																
CSC_C404						7-10												
ENG_C101											0-1					0-1		
ENG_C201				0-3														
ENG_C202						0-6												
ENG_C203				1-3														
ENG_C301									0-8									
ENG_C302						2-6												
ENG_C303										0-8								
ENG_C304				0-6														
ENG_C305																		0-11
ENG_C306									9-14									
ENG_C401								0-11										
ENG_C402											5-10					5-9		
ENG_C403			0-9															
Required BOK Hours	30	63	18	43	5	57	33	17	40	20	8	21	20	39	33	16	13	10
Required BOK Units	0-5	0-9	0-8	0-6	0-2	0-9	0-6	0-6	0-10	0-6	0-4	0-6	0-4	0-5	0-7	0-8	0-6	0-5
Elective BOK Units	6	10	9-11	7-9	3-8	10		7-11	11-14	7-10	5-10	7-10	5-8	6-8	8	9	7-9	6-11

B.6.5 Curriculum C – Course Summaries

CSC_C101: Discrete Structures for Computer Science

Review of propositional and predicate logic; methods of theorem proving; strong and weak induction; finite and infinite sets, set operations; introductions to computational complexity, theta and big-O notation; combinatorics, including permutations and combinations; discrete probability and binomial distribution.

Prerequisites: Pre-calculus or equivalent.

Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14

CE2004 BOK Coverage: CE-DSC 0-6.

CSC_C102: Programming I

Introduction to computer science with emphasis on problem solving, programming and algorithm design; use of a high-level programming language for solving problems and emphasizing program design and development; topics include basic programming constructs, expressions, conditional statements, loop statements, functions, classes and objects, data types, arrays, and strings.

Prerequisites: Pre-calculus or equivalent.

Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14

CE2004 BOK Coverage: CE-PRF 0-4; CE-SWE 0-1.

CSC_C103: Programming II

Investigate the essential properties of data structures, abstract data types, and algorithms for operating on them; to use these structures as tools to assist algorithm design; introduction of searching and sorting techniques.

Prerequisites: CSC_C102

Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14

CE2004 BOK Coverage: CE-PRF 5–8; CE-SWE 2-3.

CSC_C201: Algorithms and Data Structures

The study of representations for lists, stacks, queues, trees, and graphs; fundamental algorithms and their implementation for sorting, searching, merging, hashing, graph theoretic models, and recursive procedures.

Prerequisites: CSC_C101, CSC_C103

Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0

CE2004 BOK Coverage: CE-ALG 0–6; CE-SWE 4-5.

CSC_C202: Introduction to Computer Organization

Internal organization of computers; registers, organization, control mechanisms; instruction sets, microprogramming, hardware interfaces; datapaths and pipelining; structural, data, and branch hazards; optimization; memory and caching; non-von Neumann designs.

Prerequisites: CSC_C101, ENG_C202

Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0

CE2004 BOK Coverage: CE-CAO 0-4.

CSC_C301: Operating Systems

A study of the internal design of operating systems; topics include memory management, multiprogramming, virtual memory, paging and segmentation; job and process scheduling; multiprocessor systems; device and file management; thrashing, cache memory.

Prerequisites: CSC_C201, CSC_C202

Credit Hours: 3 Lecture Hours: 14 Lab Hours: 0 Recitation Hours: 0

CE2004 BOK Coverage: CE-DBS 0-3; CE-OPS 0-8.

CSC_C302: Computing, Ethics, and Society

Critical examination of ethical problems associated with computer technology; discussion of these problems conducted within the framework of classical philosophical ethical theories; legal and quasi-legal (i.e., policy and regulative) issues; topics addressed include the process of ethical decision-making, privacy and confidentiality, computer crime, professional codes and responsibilities, software piracy, the impact of computers on society.

Prerequisites: Sophomore standing

Credit Hours: 1 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0

CE2004 BOK Coverage: CE-HCI 0-1; CE-SPR 0-9.

CSC_C303: Computer Networks

Technical introduction to data communication; OSI reference model, layer services, protocols, LANs, packet switching and X.25, ISDN; file transfer, virtual terminals, system management and distributed processing.

Prerequisites: CSC_C103, MTH 101
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-NWK 0-8.

CSC_C304: Applied Probability and Statistics

Systematic development of the concept of probability and random process theory; topics include probability and set theory, random variables, density and distribution functions, multivariate distributions, sampling statistics and distributions, central limit theorem, estimation and the philosophy of applied statistics; application to problems in the physical sciences and engineering.

Prerequisites: MTH 102
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-PRS 0-8

CSC_C401: Computer Architecture

Study of computer architecture from classical to advanced perspectives; explores architectural characteristics of modern computer systems such as performance, instruction sets, assemblers, datapaths, pipelining, caching, memory management, I/O considerations, multiprocessing, wireless communication, and other advanced systems.

Prerequisites: CSC_C202, ENG_C202, and Senior Standing
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-CAO 3-10.

CSC_C402: Software Engineering

Study of the nature of the program development task when many people, modules and versions are involved in designing, developing and maintaining a large program or system; issues addressed include software design, specification, version control, testing, cost estimation and management; study of software systems in different domains such as database systems and HCI systems are also addressed.

Prerequisites: CSC_C201
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-DBS 0-2; CE-HCI 0-4; CE-SWE 0-9.

CSC_C403: Computer Architecture Laboratory

Experiments provide laboratory experience in the designs and operations of different types of computer architecture, memory architectures, I/O and bus subsystems, special purpose architectures, parallel processing, and distributed systems; explore hardware and software issues and tradeoffs in the design, implementation, and simulation of working computer systems.

Prerequisites: ENG_C302, CSC_C401
Credit Hours: 1 Lecture Hours: 0 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-CAO 3-5,8-9

CSC_C404: Simulation and Modeling

Fundamental principles of modeling and simulation; methodology including model formation, design of simulation experiments, analysis of generated data and validation of results; survey of applications; design project.

Prerequisites: CSC_C103
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-DIG 7-10

ENG_C101: Introduction to Engineering

Overview of the engineering profession, its genesis and evolution to the present day, including fields of engineering and career paths within same; study of ethics and with emphasis on the engineering workplace; engineering design and analysis techniques, development of problem-solving skills, communication skills; student design projects.

Prerequisites: Freshman standing
Credit Hours: 2 Lecture Hours: 28 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-HCI 0-1; CE-SPR 0-1.

ENG_C201: Engineering Circuit Analysis

Principles of linear system analysis introduced through the study of electric networks containing lumped circuit elements; DC resistive circuit analysis techniques; transient analysis with capacitors and inductors; steady-state AC analysis using phasors to study impedance and resonance.

Corequisites: MTH 201, PHY 201
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-CSG 0-3.

ENG_C202: Digital and Logic Design

Internal structure of computers; number systems and arithmetic, two's-complement arithmetic; Boolean algebra, logic design, gates, synthesis of combinatorial networks; flip-flops, registers, sequential circuits, control mechanisms, timing; data and control flow in a typical computer.

Prerequisites: CSC_C101, CSC_C103
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-DIG 0-6.

ENG_C203: Circuit Analysis Laboratory

The laboratory is designed to enhance the understanding and proper use of selected principles from circuit theory; experiments introduce basic measurement techniques and problem solving; comparisons between theoretical and experimental results are investigated in a written laboratory report; topics include meter calibration, oscilloscope use, transient and steady-state analysis, AC parallel and series circuits, electric filters, Thevenin's theorem, and operational amplifiers.

Prerequisites: ENG_C201
Credit Hours: 1 Lecture Hours: 0 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-CSG 1-3.

ENG_C301: Electronics Circuits

Principles of semiconductor electronic devices: operational amplifiers, diodes and bipolar junction transistors; amplifier specification and external characteristics; analysis of electronic circuits using graphical methods and electronic device models; analysis and design of electronic application circuits such as rectifiers, clippers, inverting amplifiers and voltage followers; introduction to digital simulators.

Prerequisites: ENG_C201
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-ELE 0-8.

ENG_C302: Digital and Logic Design Laboratory

Hands-on experience in using digital electronics by way of integrated circuits without engineering bias; practical construction, testing, and implementation of circuits useful in digital circuits and modules.

Prerequisites: ENG_C202
Credit Hours: 1 Lecture Hours: 0 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-DIG 2-6.

ENG_C303: Embedded Microprocessor Systems

Implementation of microprocessors and microcontrollers in embedded digital computer systems; topics include architecture, operations, software; hardware/software design methodology.

Prerequisites: CSC_C103, CSC_C202
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-ESY 0-8.

ENG_C304: Signals and Linear Systems

Analysis of discrete time and continuous-time signals and systems; development of Fourier analysis; determination of transfer functions and impulse response of linear systems; design of continuous-time electric filters; sampling and the Nyquist criterion; introduction of state-variable concepts.

Prerequisites: ENG_C201, MTH 301
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-CSG 0-6.

ENG_C305: VLSI Design

Study of basic methods of circuit design are presented followed by execution analysis and optimization using algorithms developed by the student; emphasis will be on structured design methodologies for MOS systems with focus on performance considerations and design methodologies for VLSI IC chips; VLSI ASIC design software packages are used to design and simulate a small CMOS chip.

Prerequisites: ENG_C301, CSC_C202
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-VLS 0-11.

ENG_C306: Electronics Laboratory

The laboratory is designed to enhance the understanding and proper use of selected principles of electronic circuits; topics cover diode and transistor applications, including feedback analysis and design, BJT and FET amplifier design and the analysis of measurement limitations of selected instruments.

Prerequisites: ENG_C301, ENG_C203
Credit Hours: 1 Lecture Hours: 0 Lab Hours: 42 Recitation Hours: 0
CE2004 BOK Coverage: CE-ELE 9-14.

ENG_C401: Digital Communication and Signal Processing

Study of basic digital communication principles; digital spectral analysis, discrete Fourier transforms, sampling and quantization; digital signal processing basics such as transformation, filtering, and basic audio and image processing concepts.

Prerequisites: ENG_C304
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-DSP 0-11.

ENG_C402: Technology and Society

The interrelationship between technology and society in the past and present is established; technological achievements of major civilizations from the Egyptians and Babylonians through the classical Mediterranean, Medieval, Renaissance, and modern industrialized eras; worldviews of different cultures toward technology are investigated, as well as both the desired and the unforeseen consequences of technological change.

Prerequisites: Junior standing
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: CE-HCI 5-10; CE-SPR 5-9.

ENG_C403: Computer Engineering Design

Integration of physical principles with mathematical analysis and/or experimental techniques; develop a basis for an individually required design project in computer engineering; design of suitable project.

Prerequisites: Senior standing
Credit Hours: 3 Lecture Hours: 14 Lab Hours: 84 Recitation Hours: 0
CE2004 BOK Coverage: CE-CSE 0-9.

B.7 Curriculum Implementation
Computer Engineering Program Representative of a Program in the United Kingdom and Other Nations

B.7.1 Program Goals and Features

This curriculum model demonstrates a typical program in computer engineering as one might find in the United Kingdom, Europe, Asia, and other parts of the world. With slight and differing modifications, the demonstrated three-year degree program would be a typical Bachelor of Engineering degree in England. The four year curriculum would be a typical Masters of Engineering degree in England or, with some slight modification, a Bachelor of Engineering degree in Scotland A heavy emphasis on technical material with little room for broader general education classes characterizes this model.

Students desiring intensive study in computer engineering will find this program to be a challenging and rewarding experience. The curriculum provides a broad foundation in the science and engineering of computers and digital systems with emphasis on theory, analysis, and design. The curriculum will also develop analytical, computer and applied skills which will enable students to analyze, design and test digital and computer systems, architectures, networks, and processes. Graduates of the program will be able to apply and evaluate various areas of computer engineering such as applied electronics, digital devices and systems, electromagnetic fields and waves, computer architectures, systems, and networks. They will also have a deep understanding of hardware issues, software issues, models, the interactions between these issues, and related applications. Graduates will possess design skills and they will have the capacity to apply their accumulating knowledge to computer systems. The thorough preparation afforded by this computer engineering curriculum includes the broad education necessary to understand the impact of engineering solutions in a global and societal context.

A combination of theory, practice, application, and attitudes accompany the construction of each module or course. The intention is to convey a certain ethos about computer engineering. Especially in the early years of a course, this is an important consideration. Any model curriculum of this kind should contain general aims (or goals) and specific objectives for the program of study; it should also capture the intended characteristics of its graduates.

For the purposes of ease of comparison, this model attempts to outline the degree in conformity with the three previous curricula implementations. This means that the curriculum comprises courses having three credit hours (or 42 contact hours) of study with approximately ten courses per year. Both three and four-year curriculum models are presented, since in many places, especially in Europe, these programs only take three years because all or most of the mathematics, science, general education experiences take place in the “gymnasium” before beginning university studies. Thus, the three-year program that follows is equivalent to an undergraduate program in many places.

B.7.2 Summary of Requirements

In these models, the introduction of concepts in computer engineering appears in the early years. The justification for this is that students really should sample the discipline they will study, a matter deemed important for motivational purposes. For the third and fourth years, the curriculum includes a number of optional classes. With regard to a four-year implementation of this model, the fourth year exposes the student with a substantial influx on new and innovative subject matter representing the current activities and developments in computer engineering. Such an approach allows students to develop specializations or specialty tracts with the intent of having a focused orientation of their studies. It also indicates that degrees in computer engineering can have different emphases and individual students can follow different specializations.

In these programs of study, one would expect that some element of laboratory experience would constitute an integral part of each course in computer engineering; the purpose of this integration is to reinforce and illustrate the work of the associated lectures. In some classes, the amount of laboratory work would typically be heavier than in other parts. Accordingly, we have adopted the following convention:

- where intensive laboratory activity is desirable, a 3-credit class is typically composed of 28 hours of lectures and 28 hours of laboratory work plus associated recitation time
- where less intensive laboratory activity is desired, then typically 14 hours of laboratory work and 42 hours of lecture work is required together with the associated recitation hours

The illustration of a program in computer engineering using this model appears in two forms: A three-year and a four-year illustration. The three-year illustration contains 30 courses and 90 credit hours of study. The four-year illustration contains 40 courses and 120 credit hours of study. The distribution of credit hours for these two illustrations of this model is as follows:

Three-year Program Credit hours	*Four-year Program* Credit hours	Topic areas
15	15	Mathematics
21	39	Required computer engineering
12	12	Required computer science
21	21	Required electrical engineering
15	15	Required software engineering
6	6	Third-year electives in computer engineering
	12	Fourth-year electives
90	*120*	*TOTAL Credit Hours for Computer Engineering Program*

The two illustrations allow for electives (options) available in the third year and the fourth year of study. Students may chose two third-year electives selected from a set of four computer engineering courses (modules). For programs following the four-year illustration, students may choose any four fourth-year electives selected from a set of fourteen computer engineering courses and four software engineering courses. The course options are as follows.

Third-year Electives (computer engineering)

Computer Graphics	Intelligent Systems and Robotics
Device Development	Multimedia Systems

Fourth-year Electives (computer engineering)

Advanced Computer Design	Computer Vision
Control Systems Engineering	Fault Tolerant Systems Design
Grid Computing	Hardware Software Co-design
High Performance Computing	Intelligent Systems
Mobile Computer Systems	Network Security
Parallel Computing and Neuro-computing	Robotics and Simulation
Safety Critical Systems	Software for Telecommunications

Fourth-year Electives (software engineering)

Computer Graphics and Multimedia	Formal Software Development
Information Systems Development	Interactive Systems Design

What follows is a representation of two possible curriculum models shown as a Three-year Illustration and a Four-year Illustration. Following these two illustrations is a mapping of courses from the three-year program to the body of knowledge for computer engineering. The four-year model is not applicable here because the fourth year generally has a specific focus more likely found in post-baccalaureate programs. The course descriptions (or modules) appear in the Course Summaries section.

B.7.3 Three-Year Curriculum Model for Curriculum D

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 1			**Semester 2**		
MTH_D101	Discrete Structures for Computing	3	MTH 103	Advanced Calculus	3
MTH_D102	Applied Probability and Statistics	3	MTH 104	Differential and Difference Equations	3
CSC_D101	Computer and Information Systems	3	CPE_D101	Concepts in Computer Engineering	3
ELE_D101	Foundations of Electronics	3	ELE_D102	Digital Circuits I	3
SWE_D101	Programming Basics	3	SWE_D102	Programming Fundamentals	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 3			**Semester 4**		
MTH 201	Mathematics for Engineers	3	CPE_D203	Operating Systems and Net-Centric Computing	3
CPE_D201	Computer Organization	3	CPE_D204	Computer Systems Engineering	3
CPE_D301	Networking & Communications	3	CSC_D202	Information Management	3
CSC_D201	Analysis and Design of Algorithms	3	ELE_D202	Analog Circuits	3
ELE_D201	Digital Circuits II	3	CPE_D302	Embedded Computer Systems	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 5			**Semester 6**		
CPE_D403	Individual Project I	3	CPE_D406	Individual Project II	3
CSC_D301	Programming Languages and Syntax-Directed Tools	3	CPE_D303	Computer Architecture	3
ELE_D301	Signals and Systems	3	ELE_D302	System Control	3
SWE_D301	Software Engineering	3	ELE_D303	Digital Signal Processing	3
	Third Year Option A	3		Third Year Option B	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*

B.7.4 Four-Year Curriculum Model for Curriculum D

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 1			**Semester 2**		
MTH_D101	Discrete Structures for Computing	3	MTH 103	Advanced Calculus	3
MTH_D102	Applied Probability and Statistics	3	MTH 104	Differential and Difference Equations	3
CSC_D101	Computer and Information Systems	3	CPE_D101	Concepts in Computer Engineering	3
ELE_D101	Foundations of Electronics	3	ELE_D102	Digital Circuits I	3
SWE_D101	Programming Basics	3	SWE_D102	Programming Fundamentals	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 3			**Semester 4**		
MTH 201	Mathematics for Engineers	3	CPE_D203	Operating Systems and Net-Centric Computing	3
CPE_D201	Computer Organization	3	CPE_D204	Computer Systems Engineering	3
CPE_D202	Professional Issues in Computer Engineering	3	CSC_D202	Information Management	3
CSC_D201	Analysis and Design of Algorithms	3	ELE_D202	Analog Circuits	3
ELE_D201	Digital Circuits II	3	SWE_D201	Building Software Systems	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 5			**Semester 6**		
CPE_D301	Networking and Communications	3	CPE_D302	Embedded Computer Systems	3
CSC_D301	Programming Languages and Syntax-Directed Tools	3	CPE_D303	Computer Architecture	3
ELE_D301	Signals and Systems	3	ELE_D302	System Control	3
SWE_D301	Software Engineering	3	ELE_D303	Digital Signal Processing	3
	Third Year Option A	3		Third Year Option B	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 7			**Semester 8**		
CPE_D401	Project Management	3	CPE_D404	Entrepreneurship	3
CPE_D402	Business and Economics of Computer Engineering	3	CPE_D405	Ubiquitous and Pervasive Computing	3
CPE_D403	Individual Project I	3	CPE_D406	Individual Project II	3
	Fourth Year Option A	3		Fourth Year Option C	3
	Fourth Year Option B	3		Fourth Year Option D	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*

B.7.5 Mapping of Computer Engineering BOK to Three-Year Curriculum D

BOK Area / Course	ALG	CAO	CSE	CSG	DBS	DIG	DSC	DSP	ELE	ESY	HCI	NWK	OPS	PRF	PRS	SPR	SWE	VLS
CPE_D101			0-9															
CPE_D201		0-4,9																
CPE_D202																0-9		
CPE_D203												0-1	0-7					
CPE_D204			0-11															
CPE_D301												0-9						
CPE_D302										0-10								
CPE_D303		5-9																
CPE_D401			7													1-8		
CPE_D402																		
CPE_D403																		
CPE_D404																		
CPE_D405												6						
CPE_D406																		
CSC_D101											0-4			0-2				
CSC_D201	0-6													3				
CSC_D202					0-8													
CSC_D301																	6-7	
ELE_D101				0-3					0-4									0-2
ELE_D102						0-6												3-5
ELE_D201						6-9			4-8									
ELE_D202									9-14									
ELE_D301				4-9				0-3										
ELE_D302				9														
ELE_D303								0-11										
SWE_D101														0-3			0-1	
SWE_D102														4-8			2-3	
SWE_D201											4-10			8			6-7	
SWE_D301																	0-9	
SWE_D302																		
MTH_D101							0-6											
MTH_D102															0-8			
Required Hours	30	63	18	43	5	57	33	17	40	20	8	21	20	39	33	16	13	10
Required Units	0-5	0-9	0-8	0-6	0-2	0-9	0-6	0-6	0-10	0-6	0-4	0-6	0-4	0-5	0-7	0-8	0-6	0-5
Elective Units	6	10	9-11	7-9	3-8	10		7-11	11-14	7-10	5-10	7-10	5-8	6-8	8	9	7-9	6-11

B.7.6 Curriculum D – Course Summaries

CPE_D101: Concepts in Computer Engineering

Range of illustrations of the applicability of developments in computer engineering exhibiting the use of hardware and software systems in a variety of different contexts including simple devices, embedded systems, systems with an important human computer interface, systems involving computer communications, and systems of a sensitive nature such as safety critical systems; issues involved in electronics, software, human computer interface, use of tools, systems, and the engineering dimension.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-CSE 0-9

CPE_D201: Computer Organization

The fundamental elements of digital logic and their use in computer construction; register level description of computer execution and the functional organization of a computer; essential elements of computer architecture; major functional components of a modern computer system. Characteristics of machine codes: instruction formats and addressing modes. The elements of machine and assembly code programming. Memory hierarchy and organization. Interfacing and communication between processor and peripheral devices. Experiments provide laboratory experience in hardware and software to interface memory and peripheral components to a computer system.

Prerequisites: CSC_D101
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-CAO 0-4, 9

CPE_D202: Professional Issues in Computer Engineering

Critical examination of ethical problems associated with computer engineering; discussion of these problems conducted within the framework of classical philosophical ethical theories; legal and quasi-legal (i.e., policy and regulative) issues; topics addressed include the process of ethical decision-making, privacy and confidentiality, computer crime, professional codes and responsibilities, professional practice, system security, impact of computers on society.

Prerequisites: Second-year standing
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14
CE2004 BOK Coverage: CE-SPR 0-9

CPE_D203: Operating Systems and Net-Centric Computing

The functionality and role of an operating system; major components, design considerations; layered approach to the design of an operating system, including the major influences on design, including high level languages, real-time issues, networking, multimedia, security; file systems, hierarchical design; process management, scheduling strategies; resource allocation strategies including memory allocation strategies; segmentation, virtual memory, cache; concurrency, synchronization principles, deadlock avoidance; operating system routines; class libraries; scripting languages, capabilities and possibilities; device management, buffering issues, interrupts, device drivers; technical issues, and in particular the software architectures, associated with net-centric computing. Basic concepts in networking and communications. Security issues.

Prerequisites: CSC_D101
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-OPS 0-7, CE-NWK 0-1.

CPE_D204: Computer Systems Engineering

Approaches to the development of systems in computer engineering; the special problems and the issues; concept of a life cycle, nature of life cycle models, phases of typical life cycles, quality issues; process and process improvement; issues of teams, team selection, roles in teams, elements of team work; selection of support tools, standards and technologies; techniques and approaches associated with the different phases; special problems of design and the issues associated with tradeoffs, special problem of hardware/software tradeoffs; testing; maintenance; project management.

Prerequisites: CPE_D101
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 28 Recitation Hours: 14
CE2004 BOK Coverage: CE-CSE 0-11

CPE_D301: Networking and Communications

Computers and computer communication; problems of security, reliability; speeds, capacity measures, reliability measures; physical realities and the limitations; wireless possibilities; communications network architectures, computer network protocols; variants on the basic topologies; local and wide area networks; client server computing; data integrity and data security, problems and solutions; performance issues; network management; nature and special problems of mobile computing.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-NWK 0-9

CPE_D302: Embedded Computer Systems

Nature of embedded systems, particular problems, special issues; role in computer engineering; embedded microcontrollers, embedded software; real time systems, problems of timing and scheduling; testing and performance issues, reliability; low power computing, energy sources, leakage; design methodologies, software tool support for development of such systems; problems of maintenance and upgrade; networked embedded systems.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 28 Recitation Hours: 14
CE2004 BOK Coverage: CE-ESY 0-10

CPE_D303: Computer Architecture

Design principles associated with modern computer architectures; performance and cost considerations; architectural features influenced by such features as operating systems and window systems, high level languages, networking, security considerations; processor implementation strategies, micro-programming, pipelining, CISC and RISC, vector processors; memory hierarchy, cache, virtual memory organization for high performance machines; special purpose components and devices; simple demonstrations provide experience in the designs and operations of different types of computer architecture such as memory architectures, I/O and bus subsystems, special purpose architectures, parallel processing, and distributed systems; explore hardware and software issues and tradeoffs in the design, implementation, and simulation of working computer systems.

Prerequisites: CPE_D201
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-CAO 5-9

CPE_D401: Project Management in Computer Engineering

Project initiation, management, and success, appraisal and risk, quality systems and implementation, environmental impacts; contracts, costs, finance, planning, organization, personnel management; contract strategies and policy; turnkey operations, global issues, international commerce, negotiation, customs, and law.

Prerequisites: Fourth-year standing
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-CSE 7, CE-SPR 1-8.

CPE_D402: Business and Economics of Computer Engineering

Engineering and uncertainty, engineering processes, strategies, proposals, decision making; economic concepts, utility, value, cost, consumers; supply and demand; initial costs, maintenance, fixed, variable, and marginal costs; interest rates, simple and compound interest; money value, past, present, and future values; cash flow; present and future worth, payback periods.

Prerequisites: Fourth-year standing
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: None.

CPE_D403: Individual Project I

Comprehensive project spanning two semesters; students undertake an individual project which involves addressing a significant technical problem they embark on this under the guidance of a supervisor; student are expected to demonstrate an ability to apply the disciplined approaches of the course in addressing the solution to the problem; students produce a final thesis on the work and this together with a demonstration of the working system will form the assessment.

Prerequisites: Fourth-year standing
Credit Hours: 3 Lecture Hours: 3 Lab Hours: 56 Recitation Hours: 14
CE2004 BOK Coverage: Project dependent.

CPE_D404: Entrepreneurship and the Engineer

Discusses basic concepts of marketing, business organization, management accounting, business finance, and financial feasibility analysis of new business ventures and of new project proposals in established firms; appreciate the financial risks and rewards; strategies for investing in new ventures; entrepreneurial strategies, venture development processes, bringing products from the idea to market and operation; business planning, implementation, operation, and success, business plans, organization, budgets, accounting methods and processes, capital and debt, business analyses.

Prerequisites: Fourth-year standing
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 0
CE2004 BOK Coverage: None.

CPE_D405: Ubiquitous and Pervasive Computing

Discusses current trends towards universal presence of mobile computing, computer networks, and wireless communication; how network devices are aware of their environment; identification of current status, fundamental issues, future problems and applications; current research topics in the area of ubiquitous and pervasive computing; design issues; integration and processing of sensor-based input; wireless infrastructures; security and user-interfaces; integrated, multimodal input and output and application areas.

Prerequisites: Fourth-year standing
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-NWK 6

CPE_D406: Individual Project II

Continuation of Individual Project I.

Prerequisites: Fourth-year standing
Credit Hours: 3 Lecture Hours: 0 Lab Hours: 42 Recitation Hours: 42
CE2004 BOK Coverage: Project dependent.

CSC_D101: Computer and Information Systems

Representation of data of different kinds; elements of machine code and assembly language coding; role and function of an operating system (including networking, e-mail and distributed systems) and the associated functionality; programming language level, facilities and libraries; applications including description of the functionality of the relevant software (word processors, databases, spreadsheets) and their use; human interaction, importance and relevance of interface software; elements of computer interaction including desirable properties of screen design and interfaces; fundamentals of the web; use of browsers and search engines in information retrieval; simple web page construction; illustrations of information servers; search strategies; information storage and retrieval; legal issues of copyright and intellectual property rights.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-HCI 0-4, CE-PRF 0-2.

CSC_D201: Analysis and Design of Algorithms

Elementary ideas and results on discrete probability; mathematical foundations needed to support measures of complexity and performance; basic concepts from counting; concepts of graphs and trees; basic strategies that underpin the design of algorithms; fundamental algorithms for counting, searching, sorting, manipulation of hash tables, symbol tables, queues, trees, and graphs; distributed algorithms for certain simple tasks; fundamentals of computability theory; relevance to security; relevance of design and analysis of algorithms to software design and implementation.

Prerequisites: MTH_D101, SWE_D102
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 28 Recitation Hours: 14
CE2004 BOK Coverage: CE-ALG 0-6, CE-PRF 3

CSC_D202: Information Management

Relevance of information management in the context of computer engineering; introduction to database systems and the relational model; normal forms and their benefits; building databases, underlying methodology, database languages; issues associated with information retrieval; SQL, its use and power; information systems in the context of networks, intranets, extranets; special systems and applications; particular issues, access, security, and integrity; relevant legal and ethical issues.

Prerequisites: SWE_D102
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-DBS 0-8

CSC_D301: Programming Languages and Syntax-Directed Tools

History of the development of languages; different flavors of languages, programming, scripting, mark-up, specification; language role, characteristics, comparisons; different programming paradigms, significance, main areas of application, imperative, functional, logic, object-oriented languages; concurrency; aims and objectives of language design; principles of language design, including limitations; interaction between language design and the translation process; basic approaches to translation; aims and objectives of translation; major components of translation and their implementation; library design, separate compilation, design considerations, and implementation.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 28 Recitation Hours: 14
CE2004 BOK Coverage: CE-SWE 6-7.

ELE_D101: Foundations of Electronics

Introduction to basic electrical quantities such as charge, current, voltage, energy and power. Introduction to classical dynamics, electrostatics, and magnetism. Basic laws such as Kirchoff's law, Ohm's law; Thevenin's theorem, Norton's theorem. Resistive circuits and networks, reactive circuits and networks. Capacitance, inductance, damping, transformers. Electronic properties of materials. Diodes and diode circuits. MOS transistors and biasing. MOS logic families.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-CSG 0-3, CE-ELE 0-4, CE-VLS 0-2.

ELE_D102: Digital Circuits I

Basic switching theory, combinational logic circuits; modular design of combinational circuits; memory elements; sequential logic circuits; digital systems design; understanding and analysis of the basic types of circuits and electrical networks as used in electronics, communications, and power applications.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-DIG 0-6, CE-VLS 3-5.

ELE_D201: Digital Circuits II

Review of MOS families and circuits; bipolar transistors and logic families; digital parameters and issues; storage elements; interfacing logic families and standard busses; fundamentals of digital systems design including state diagrams; modeling and simulation, use of relevant tools; use of CAD tools; design carried out for testability and for other such characteristics; problems of verification and validation; formal verification.

Prerequisites: ELE_D102
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 28 Recitation Hours: 14
CE2004 BOK Coverage: CE-DIG 6-9, CE-ELE 4-8.

ELE_D202: Analog Circuits

Data conversion issues, A/D and D/A circuits; electronic voltage and current sources; low and high pass filters, Chebyshev and Butterworth approximations, Sallen-Key; negative feedback; operational amplifier circuits; introduction to bipolar junction transistors.

Prerequisites: ELE_D201
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-ELE 9-14.

ELE_D301: Signals and Systems

The concept of signals and systems, both continuous and discrete-time; signal manipulation; signal symmetry and orthogonality; system linearity and time invariants; system impulse response and step response; frequency response, sinusoidal analysis, convolution, and correlation; sampling in time and quantizing in amplitude; Laplace transform; Fourier analysis, filters; analysis of discrete time signals and systems using z-transforms; inverse transformation procedures.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-CSG 4-9, CE-DSP 0-3.

ELE_D302: System Control

Review of complex numbers, superposition, compound systems; frequency domain representation; Laplace transform representation; system representation in time domain; first and second order systems; damping, stability, poles, and zeros; feedback block diagrams; open loop and closed loop systems; steady state error; introduction to Bode plots and Bode plot analysis introduction to proportional control.

Prerequisites: ELE_D301
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-CSG 9.

ELE_D303: Digital Signal Processing

Purpose of digital signal processing (DSP), theories and concepts, role of DSP in the context of computer engineering; analysis of digital spectra; application of discrete Fourier transforms, convolution types; filtering, digital filtering; transforms; discrete time signals; sampling issues; applications to include image processing, audio processing; use of relevant software tools.

Prerequisites: ELE_D301
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-DSP 0-11

SWE_D101: Programming Basics

Introduction to the concepts of requirements and specification; basic concepts associated with programming languages and their translation; elementary programming, primitive data types, operations, simple language constructs; simple algorithms and problem solving involving counting, scanning elements, selecting elements (such as maxima and minima), iteration; use of arrays, strings and simple pre-defined classes; routines or methods as a fundamental abstraction mechanism; principles associated with and the design and construction of these; use of simple libraries, classes; simple aspects of quality of software; the related activities of software testing and validation.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 28 Recitation Hours: 14
CE2004 BOK Coverage: CE-PRF 0-3, CE-SWE 0-1.

SWE_D102: Programming Fundamentals

Concepts from predicate logic; ideas from object-oriented programming, methods, classes, information hiding, and inheritance; fundamental algorithms, sorting and searching; fundamental data structures, linked data structures, user defined classes; concept of recursion, benefits and problems; exception handling; using APIs; simple graphics programming; concept of software design.

Prerequisites: SWE_D101
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 28 Recitation Hours: 14
CE2004 BOK Coverage: CE-PRF 4-8, CE-SWE 2-3.

SWE_D201: Building Software Systems

Concepts of open source, shareware, freeware; issues of quality, conditions of use, availability; issues of software reuse; program libraries, software components; creation of additional libraries and other components; application program interfaces; use of separate compilations; use of software libraries and other software components; problems of building large systems; assessment of software including interfaces such as metrics and measures; criteria; simple principles of interface design; multimedia issues; special problems associated with color, sound, video and multimedia; advanced issues in object-oriented programming, modularity, storage management issues, parallelism; client server computing, different kinds of servers, the role of middleware; overview of the software support needed for client services and server services; illustrations of the use of object oriented techniques applied to the building of certain commonly used software tools; applets and servelets; simple design patterns; nature of the software life cycle and its different phases; concept of process; differences across various developments and the reasons for the differences.

Prerequisites: SWE_D102
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 28 Recitation Hours: 14
CE2004 BOK Coverage: CE-HCI 4-10, CE-PRF 8, CE-SWE 6-7.

SWE_D301: Software Engineering

Software engineering, role of software engineers; evaluation of software and principles thereof, software lifecycle models; notions of requirements, specification, design implementation; main techniques; important of maintenance; quality concerns at all stages of the software development process; concept of process; software process maturity models; software process improvement; aspects of software engineering, important benefits of and good practice in software re-use; verification and validation; the use of metrics; selection of and use of tools; the nature and structure of teams; human computer interface as a software engineering activity; related life cycles; standards; use of relevant libraries; importance of practical activity; group activity as an important skill for these engineers.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 28 Recitation Hours: 14
CE2004 BOK Coverage: CE-SWE 0-9.

MTH_D101: Discrete Structures for Computing

Basic mathematical notions of sets, relations, and functions, and operations involving the same; logic and its role, propositional logic, truth tables, issues of equivalence, limitations; predicate logic, its power and its limitations, relevance in the context of computer engineering; proof techniques; commonly occurring mathematical concepts such as graphs, trees; representational issues; relevance of these to computer engineering; recursion; counting; combinatorics; relevance of these ideas to computer engineering.

Prerequisites: One course in calculus
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0 Recitation Hours: 14
CE2004 BOK Coverage: CE-DSC 0-6

MTH_D102: Applied Probability and Statistics

Randomness, finite probability space, probability measure, events; conditional probability, independence, Bayes' theorem; discrete random variables; binomial and Poisson distributions; concepts of mean and variance; continuous random variables; exponential and normal distribution, probability density functions, calculation of mean and variance; central limit theorem and the implications for the normal distribution; purpose and the nature of sampling; nature of estimates, point estimates, interval estimates; maximum likelihood principle approach, least squares approach; confidence intervals; estimates for one or two samples; development of models and associated hypotheses; nature of hypothesis formulation, null and alternate hypotheses, testing hypotheses; criteria for acceptance of hypothesis t-test, chi-squared test; correlation and regression; Markov processes, discrete time systems and continuous time systems; queuing theory including system simulation and modeling, queuing methods; use of appropriate statistical packages.

Prerequisites: Two courses in calculus and two courses in physics
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 14 Recitation Hours: 14
CE2004 BOK Coverage: CE-PRS 0-8.