Computer Engineering Curricula 2016

CE2016

Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering

2016 December 15

A Report in the Computing Curricula Series

Joint Task Force on Computer Engineering Curricula

Association for Computing Machinery (ACM)
IEEE Computer Society

Association for Computing Machinery

Computer Engineering

Curricula 2016

Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering

A Report in the Computing Curricula Series

Joint Task Group on Computer Engineering Curricula

Association for Computing Machinery (ACM)
IEEE Computer Society

2016 December 15

ISBN: 978-1-4503-4875-1
DOI: 10.1145/3025098
Web link: https://dx.doi.org/10.1145/3025098
ACM Order Number: 999163

When available, you may order additional copies from:

ACM Order Department
P.O. Box 30777
New York, NY 10087-0777

+1-800-342-6626
+1-212-626-0500 (outside U.S.)
orders@acm.org

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros
P.O. Box 3014
Los Alamitos, CA 90720-1314

Tel: +1 800 272 6657
Fax: +1 714 821 4641
http://computer.org/cspress
csbook@computer.org

Sponsoring Societies

This report was made possible by
financial support from the following societies:

Association for Computing Machinery (ACM)
IEEE Computer Society

The CE2016 Final Report has been endorsed by
Association for Computing Machinery (ACM) and the IEEE Computer Society.

Cover art by Robert Vizzini
Printed in the United States of America

Computer Engineering

Curricula 2016

Final Report

2016 December 15

A Report in the Computing Curricula Series

Joint Task Group on Computer Engineering Curricula

Association for Computing Machinery (ACM)
IEEE Computer Society

CE2016 Steering Committee

Contents

Executive Summary

This report presents curriculum guidelines for undergraduate degree programs in computer engineering. It draws upon the 2004 published curricular report in computer engineering titled, *Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering*, also known as CE2004. This report also draws upon recent efforts in computing curricula developed by the Association for Computing Machinery (ACM), the IEEE Computer Society, and the Association for Information Systems (AIS). These efforts resulted in published curricula recommendations in computer science [ACM/IEEECS, 2013],[1] information systems [ACM/AIS, 2010], information technology [ACM/IEEECS, 2008], and software engineering [ACM/IEEECS, 2015]. New curricula recommendations for information technology, as of this writing, are expected to be published in 2017.

Computer engineering as an academic field encompasses the broad areas of electrical or electronics engineering and computer science. We define computer engineering 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.*

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

This report provides some background on the field of computer engineering 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. The report also describes the expected background, knowledge, and skills employers expect to see from graduates of computer engineering programs. These expectations include the ability to design computer systems, the realization of the importance of practicing as professionals, and the breadth and depth of knowledge expected of a practicing engineer. The report also discusses ways in which programs in computer engineering may have to stand up to the scrutiny of validation and accreditation by government or private agencies.

The foundation for this report is a fundamental body of knowledge from which an institution can develop or modify a curriculum to fit its needs. This body of knowledge, also known as BoK, contains broad knowledge areas (KAs) that are applicable to all computer engineering programs worldwide. Each knowledge area comprises a thematic scope and a set of knowledge units (KUs). A set of learning outcomes defines each knowledge unit. The report further identifies some knowledge units as “core” that should appear in every implemented curriculum; the remaining knowledge units are supplementary. Core units represent the minimal knowledge or depth a program should cover in each knowledge area. A curriculum in computer engineering that contains only core units would be very incomplete.

A computer engineering program should contain sufficient coursework at the introductory, intermediate, and advanced levels based on the body of knowledge for 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 necessary to this discipline. A design component is vital to the program and it typically culminates with a capstone or senior project experience. 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, personal (soft) skills, oral and written communication skills,

[1] The full references for these bracketed citations are found immediately following Appendix D.

teamwork, and a variety of laboratory experiences are fundamental to the study of computer engineering. Additionally, the report includes sample curricula models that illustrate methodologies institutions might select to develop curricula in computer engineering based on their locale, mission, and specific student goals.

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 global 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.

— CE2016 Steering Committee

Chapter 1

Introduction

In the 1980s, the Association for Computing Machinery (ACM) and the Computer Society of the Institute for Electrical and Electronics Engineers (IEEE-CS) established a joint committee to develop computing curricula (CC) guidelines for undergraduate degree programs in computing. This effort resulted into Computing Curricula 1991, also called CC1991 or CC'91 [CC91]. Over the years, this effort resulted in a series of documents whose development remains ongoing. One of the documents that emerged for the efforts from CC'91 was Computing Curricula Guidelines for Computer Engineering Programs, also known as CE2004 [ACM/IEEECS, 2004]. The report presented here—referred to as CE2016—focuses specifically on computer engineering and is an update of the CE2004 report.

One goal of the CE2016 effort is to revise CE2004 so that it incorporates the developments of the past decade and the projected needs of the decade to come. Computing technologies have developed rapidly over that time in ways that have had a profound effect on curriculum design and pedagogy. Another goal of this effort includes supporting a group of professionals who are responsible for developing and teaching a range of degree programs in computer engineering worldwide. Hence, this report must provide international perspectives and reflect a global view of computing related developments in computer engineering.

1.1 Overall Structure of the Computing Curricula Project

Due to the broadening scope of computing—and the feedback received on prior publications—the CC initiative contains several reports. These reports describe separately vital areas such as computer engineering, computer science, information systems, information technology, and software engineering, each with its own identity and pedagogical traditions. To encompass the different disciplines that are part of the overall scope of computing, professional organizations have undertaken similar reports in five curricular areas. These areas include computer engineering (2004), computer science (2001, 2008, 2013), information systems (1997, 2002, 2006, 2010), information technology (2008), and software engineering (2004, 2015). The references for the most recent examples of each of these are [ACM/IEEECS,2004], [ACM/IEEECS,2013], [ACM/IEEECS,2010], [ACM/IEEECS,2008], [ACM/IEEECS,2015].

As the individual reports unfold to completion, representatives from the five computing disciplines produced an overview report (2005) 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 is shown in Figure 1.1—derived from Figure 1.1 in the overview report [OVERVIEW].

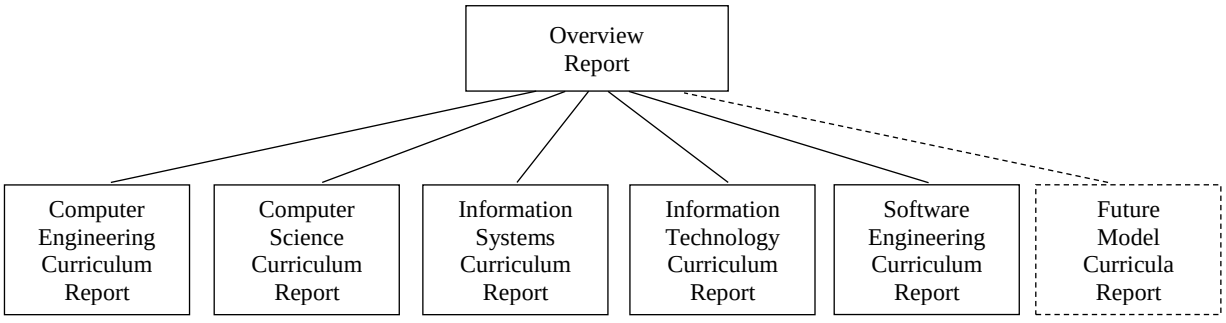

Figure 1.1: Computing curricula reports

Future curricula reports are currently under discussion. Two of those under early discussion are cybersecurity and data science.

ACM, IEEE-CS, and other professional societies and organizations have given individual groups the freedom to produce reports that best reflect the needs and requirements of their specific disciplines. However, they did request that groups address a certain minimal number of matters and, consequently, include certain components in their individual reports. The minimal set includes:

- a body of knowledge (BoK) for the field
- a set of courses that cover the body of knowledge in one or more ways
- core requirements for the discipline that apply to all undergraduates, and
- the characteristics of graduates of degree programs

Professional organizations viewed the set of requirements as minimal in order to avoid being prescriptive. Experts on curricular teams have had and still have the freedom to act independently, but reports must have this commonality among them. The anticipation is that within each discipline, reports will exceed this minimal set in various ways.

1.2 Overview of the CE2016 Process

In response to the challenges of the previously mentioned mandate for CE2016, ACM and IEEE-CS formed a steering committee to address curricular interests in computer engineering. In discharging its duty, this committee felt that it was vital to involve the wider community; hence, the committee consists of representatives from China, Scotland, and the United States. Additionally, several consultative activities have occurred worldwide to confirm the global view expressed in this volume.

The recommendations in this report are the primary responsibility of the CE2016 steering committee; the membership appears at the beginning of this report. Given the scale of the CE2016 project and the scope over which it extends, it was necessary to secure the involvement of many additional experts, representing a wide range of constituencies and areas of expertise. The list of these contributing reviewers from throughout the world who made significant contributions or commented on improving this report is found in Appendix D.

1.3 Underlying Principles

Computer engineering is a growing and important area of endeavor. The CE2016 steering committee established a set of underlying principles to guide its work. The presentation here is not in order of priority.

1. 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.
2. The 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.
3. It is important to seek and identify the fundamental skills and knowledge that all computer engineering graduates must possess. Computer engineering is a broad-based discipline. The final report must identify the common concepts and skills of the discipline.
4. The required core of the body of knowledge should be as small as reasonably possible. It is important to keep the size of the core to a minimum to allow flexibility, customization, and choice in other parts of the curriculum to enable creation of individualized programs.

5. Computer engineering must include appropriate and necessary design and laboratory experiences. A computer engineering program should include “hands-on” experiences in designing, building, and testing both hardware and software systems.
6. Computer engineering curricula are often subject to accreditation, licensure, or governmental constraints. This report recognizes these existing external constraints and provides guidance for their evolution.
7. A 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.
8. This 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.
9. The development of the final report must encompass 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.
10. This computer engineering report must strive to be international in scope. Despite the fact that curricular requirements differ from country to country, this report must be useful for computing educators throughout the world. Although educational practice in the United States may influence curriculum, the report should make 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.
11. Relevant tools, standards, and/or engineering constraints should appear throughout the body of knowledge since this principle is fundamental to all practicing engineers.
12. Learning outcomes are a necessary component in undergraduate professional education. These learning outcomes reflect inherent knowledge and concepts. Hence, ‘topics’ are redundant and should not be included as part of this report.
13. Integration of hardware-software systems is critical to the work of computer engineering. Since computer systems include integrated hardware and software components, this report should emphasize the development of a “whole computer” or a “complete computer” in the laboratory experiences that include exposure to hardware, operating systems, and software systems in the context of relevant applications.
14. The concept of design must be a recurring theme throughout the report.

1.4 Structure of the CE2016 Report

This CE2016 report addresses undergraduate programs in computer engineering. The main body of the report consists of six chapters in addition to this one. Chapter 2 illustrates how computer engineering has evolved as a discipline. It also highlights many of the characteristics expected of computer engineering graduates, 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. Chapters 3 and 4 present overviews of the computer engineering body of knowledge and describe curriculum recommendations. These chapters also articulate learning outcomes, the differences between core and elective or supplementary knowledge units, the number of core hours in the program, the importance of design and laboratory experiences, and various skills individuals need to become effective computer engineers. Chapter 5 highlights the importance of professionalism in the practice of computer engineering. Chapter 6 provides a discussion of issues affecting the implementation of a computer engineering curriculum. These include the arrangement of the student’s program of study, including courses within the major and those in other components of the educational experience as well as other implementation considerations. Chapter 7 suggests some challenges that may arise when creating or continuing computer engineering programs. This report also provides a list of cited references.

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 student outcomes. Appendix B illustrates sample curricula and typical

catalog course descriptions as they might appear at different academic institutions. The steering committee is hopeful that providing the body of knowledge, sample curricula, and course descriptions will help departments create effective curricula or help them improve the curricula they already have. Appendix C provides guidance in developing laboratories associated with a computer engineering program. Appendix D acknowledges external reviewers from throughout the world who have made significant contributions or have commented on improving this report.

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, showing how it evolved over time. It also highlights expected preparation for entering the curriculum, some of the characteristics expected from its graduates, and student outcomes and assessment. The chapter also highlights for graduates the importance of having a proper sense of professionalism to ensure a proper perspective in the practice of computer engineering.

2.1 Background

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, computer-controlled equipment, and networks of intelligent devices. Traditionally, computer engineering is some combination of both electrical engineering (EE) and computer science (CS). It has evolved over the past four decades as a separate discipline, although intimately related to computer science and electrical engineering. 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 fact, 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 and complete systems on chips have created an environment in which computers have become pervasive and replaced more conventional electronic devices. These applications manifest themselves in the proliferation of mobile smart phones, tablet computers, multimedia and location-aware devices, wireless networks, and similar products. Computer engineering also reveals itself in the myriad of applications involving embedded systems, namely those computing systems that appear in applications such as automobiles, control systems, major appliances, and the internet of things.

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, automobile, 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. Computer engineers also work on distributed computing environments—local and wide area networks, wireless networks, internets, intranets—and embedded computer systems—such as in aircraft, spacecraft, and automobile control systems where they 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 multimedia, computer, and networking technologies) resulting in widespread and ready access to information on an enormous scale. This convergence of technologies and the associated innovation lie at the heart of economic development and the future of many organizations, creating many opportunities and challenges for computer engineers. 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 electrical engineering programs, extending digital logic design to the creation of small-scale digital systems and, eventually, to the design of microprocessors and computer systems. Later, curricula in computer engineering increasingly began to include and finally evolved to integrate relevant knowledge areas from computer science. Today, that trend is diminishing and CE programs reflect their own knowledge areas. This CE2016 report reflects the new approach.

In China, computing education in universities has developed over the last sixty years. The first fast developing stage began at the end of 1950s. Fifteen universities set up disciplines concerning computers. Most of these disciplines at that time went by the name 'computing equipment' to emphasize design of computers from fundamental components. Their requirements were like the requirements of computer engineering. The second fast developing stage was from the end of 1970s to the middle of 1980s. Seventy-four disciplines were set up during this period; their names were 'Computer and Application' or 'Computer Software.' Some universities placed more emphasis on computer hardware for the 'Computer and Application' discipline, reflecting the demand on computer engineering. From the middle 1990s, Chinese computer education entered the third fast developing stage, with over five hundred disciplines set up in the universities. In this stage, the Ministry of Education specified 'computer science and technology' as the first-level discipline that included 'computer software and theory,' 'computer systems organization,' and 'computer applied technology' as the second-level disciplines. The knowledge of computer engineering occurred in the first-level discipline, 'computer science and technology.' After 2010, the Ministry of Education added 'software engineering' and 'cyberspace security' as new first-level disciplines, which were as important as computer science and technology.

Within the UK, the term computer engineering tended to be associated with work stemming from the universities of Manchester and Cambridge where some of the early computers were developed. The University of Manchester launched a degree in computer engineering in 1980, but subsequently they discontinued it. Prior to 1980, classes or modules addressing subjects in computer engineering did exist at several universities. Nowadays degree courses in "computer systems engineering" or "computing and electronic systems" are far more common and they reflect an attention to wider engineering subjects that subsume computer engineering. Almost all universities within the UK offer a degree program in the general area of computer systems engineering. Due largely to marketing forces, there are many degree titles, some of which reflect a period in industry (e.g., computer systems engineering with industrial placement) or the inclusion of a modern language (e.g., computer systems engineering with a modern language).

The Engineering Council [EngC] has overall responsibility for the accreditation of engineering degrees in the UK and maintains a list of accredited degree programs. Its search facility reveals that as of August 2015 the Engineering Council has accredited some 44 computer systems engineering degree programs and 11 computer and electronic systems engineering degree programs. Due to the great variety of degree titles in the UK, we should regard these as underestimates of accredited programs.

In the United States, the first computer engineering program accredited by the Engineering Accreditation Commission (EAC) of ABET (formerly the Accreditation Board for Engineering and Technology) was at Case Western Reserve University in 1971. As of October 2016, the EAC of ABET has accredited over 301 computer engineering or similarly named programs, including 49 programs outside of the United States. 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 370 accredited electrical or electronics engineering programs.

One expects that the growth trend in computer engineering will increase as computing and electronic technologies become more complex. The evolution may take many forms, including but not limited to:

- an expanded content from and tighter integration with computer science

- collaboration with the software engineering discipline on application-focused projects and embedded systems with a greater emphasis on design and analysis tools to manage complexity
- a re-integration with electrical engineering, as computer-based systems becomes dominant in areas such as control systems and telecommunications

Table 2.1: Summary of currently ABET-accredited baccalaureate level computer engineering programs (as of October 2016; number of international programs in parentheses)

Program Name	Year of Initial Accreditation					Totals
	Before 1980	1980 - 1989	1990 - 1999	2000 - 2009	2010 - 2016	
Computer Engineering	10	39	56	96 (16)	40 (21)	241 (37)
Computer Systems Engineering	2	1	0	3 (1)	2 (1)	8 (2)
Electrical and Computer Engineering	8	5	3	12 (3)	4 (1)	32 (4)
Computer Science and Engineering	2	6	2	3 (1)	3 (1)	16 (2)
Other titles	0	0	0	1 (1)	3 (3)	4 (4)
Total	22	51	61	115 (22)	52 (27)	301 (49)

2.3 Characteristics of computer engineering graduates

With the ubiquity of computers, computer-based systems, and networks in the world today, computer engineers must be versatile in the knowledge drawn from standard study areas in computer science and electrical engineering as well as the foundations in mathematics and sciences. The rapid pace of change in the computing field requires that computer engineers be lifelong 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 possess the following three characteristics:

- the ability to design computers, computer-based systems, and networks that include both hardware and software as well as their integration 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
- a breadth of knowledge in mathematics and engineering sciences, associated with the broader scope of engineering and beyond that narrowly required for the field
- acquisition and maintenance of a preparation for professional practice in engineering

Other related disciplines can be described as follows.

- 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 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 a level of responsibility in computer engineers because the systems they design (such as 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 understand 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 refrain from activities that would tarnish their image or that of their practicing colleagues. Because of the importance of professionalism, Chapter 5 is devoted to an expanded discussion of professional practice and responsibilities.

2.3.3 Ability to design

Engineering draws heavily on the ability to design. The International Technology and Engineering Educators Association (ITEEA) 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.” [ITEEA] 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. Novel ideas, of course, are especially important.

Design is fundamental to all engineering. For the computer engineer, design relates to the creation and integration of 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 and integrate 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 4 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, control systems, and integrated circuit design. Foundational areas 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 subjects 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 areas 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 3, 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. From a higher-level perspective, however, one can reasonably expect several characteristics of all computer engineering graduates. These include the following.

- *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, analyzing, and maintaining it over the lifetime of the system. They should understand its operation 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 subject areas 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 and their integration, building on prior work, and including at least one major project.
- *Use of Tools*—Graduates should be able to use 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—within an electrical engineering department, within an electrical and computer engineering department or within a combined 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. Finally, computer engineering programs can be jointly administered by two such departments (e.g., electrical engineering and computer science). In such cases, the programs attempt to strike a balance to integrate the hardware and software components of the curriculum.

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 understand the 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. Specifically, 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.

In Australia, professional accreditation of entry to practice engineering programs is the responsibility of Engineers Australia, and normally occurs on a five-yearly cycle. Accreditation ensures academic institutions consistently meet national and international benchmarks, and engineering graduates of an accredited program receive membership to Engineers Australia at the relevant career grade, and enjoy reciprocal privileges by equivalent professional bodies overseas. Engineers Australia uses an Australian Qualifications Framework (AQF) that came into effect in 2015.

In the People’s Republic of China, the Ministry of Education organized three evaluations—in 2004, 2008 and 2012. The evaluations provided services whereby universities could choose to join the evaluation. The evaluation held in 2012 had participation by 391 universities and institutions that covered 4235 disciplines. Almost all “211 Project” universities and “985 Project” universities, with only two missing universities, participated in the evaluation. The percent joining the state-level key disciplines reached 93%. Computer science and technology, which includes computer engineering, is one of the first-level disciplines in the evaluation. The evaluation indexes include teachers and resources, research quality, training quality, and discipline reputation. The evaluation lasts over one year. The ministry published the evaluation reports in January of 2013. The Ministry of Education is now preparing for the fourth evaluation cycle.

In the United Kingdom, benchmarking of degrees has occurred as part of governmental quality assurance efforts. Each institution must demonstrate that their degrees meet the requisite benchmark standards for that discipline.

One example of these benchmark standards is the subject benchmark statement [SBS]. Benchmarking statements typically define both threshold (minimal) and modal (average) expectations with respect to demonstrated student knowledge, skills, and judgment. The Engineering Council (EngC) has overall responsibility for the accreditation of engineering degree programs within the United Kingdom and beyond. 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 [EngC]. In general, the British Computer Society (BCS) carries out accreditation of computing degree programs. Either the BCS or the Institution of Engineering and Technology (IET) can accredit degree programs in computer engineering. Joint accreditation by both societies is common.

Within the United States, ABET accreditation is widely recognized and accepted. In addition, ABET currently accredits programs in twenty-eight other countries. The ABET EAC Criteria for Accrediting Engineering Programs [ABET, 2016] are intended to ensure that all ABET EAC-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 must demonstrate that all graduates achieve a set of student 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.

Many countries have established their own processes for evaluation and/or accreditation through governmental or professional societies. Mutual recognition of the evaluation and/or accreditation process exists through the mechanisms of the Washington Accord [Washington], the Seoul Accord [Seoul], the Sydney Accord [Sydney], the Dublin Accord [Dublin], European Federation of National Engineering Associations [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 programs in computer engineering will also 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

The Computing Engineering Body of Knowledge

A curriculum for undergraduate study in computer engineering (CE) should reflect the current needs of computer engineering students as well as prospects for graduate study and employment in the workplace. The curriculum guidelines should also reflect current educational practice and suggest improvements where necessary. The discussion that follows attempts to accomplish this in proposing a body of knowledge commensurate with producing competent computer engineering graduates.

This report defines a collection of knowledge areas (KAs) comprising associated core and supplementary knowledge units (KUs) with associated learning outcomes. Collectively, the set of knowledge areas forms the basis of a curriculum but they are not sufficient by themselves. The knowledge areas presented in this report represent approximately 50% of the discipline-specific content of a typical four-year curriculum. Additionally, CE programs should ensure that the discipline-specific content comprises at least 1.5 years of engineering topics. The remaining chapters of this report address other issues related to the design of a complete computer engineering curriculum that meets the goals of a specific program and institution.

3.1 Structure of the body of knowledge

The CE body of knowledge has a three-level hierarchical structure. The highest level of the hierarchy is the knowledge area that represents a specific disciplinary subfield, *not* a course. Knowledge areas contain an "area scope" that describes the context of the specific knowledge area. The knowledge areas are broken down into smaller divisions called knowledge units (KUs) that represent individual themes within an area. A set of learning outcomes, representing the lowest level of the hierarchy, then describes each knowledge unit.

3.1.1 Core and supplementary components

One of the goals in updating the CE2004 report was to keep the required component of the body of knowledge as small as possible. This was done to allow programs in computer engineering to be as flexible as possible since program goals or objectives vary widely from program to program. To implement this principle, there is a distinction among the KUs to differentiate the core or essential units to the curriculum from those that are supplementary or extra units. Core components comprise knowledge and skills for which there is broad consensus that anyone obtaining a four-year degree in the field should acquire. Supplementary components comprise knowledge and skills that reflect expectations for additional work, conforming to the needs of a program.

In discussing the CE2016 recommendations, the steering committee found it helpful to emphasize the following points.

- The core components refer to the knowledge and skills that *every* student in *all* computer engineering degree programs should attain. Several learning outcomes that are important in the education of many students are not included as core and appear as supplementary. Absence of some learning outcomes among the core components does not imply a negative judgment about their value, importance, or relevance. Rather, it simply means that the learning outcome is not a requirement of *every* student in *all* CE degree programs.
- The knowledge areas are *not* courses and the core components do *not* constitute a complete curriculum. Each program may choose to cover the core knowledge units in a variety of ways.
- Additional technical areas, as well as supporting mathematics, science, and general studies, are necessary

to produce a competent computer engineer.

- It is not the case that a program should achieve core knowledge units only within a set of introductory courses early in the four-year curriculum. While some core knowledge units are introductory, a program can address some core KUs only after students have developed significant background in their studies.

3.1.2 Assessing the time required to cover a unit

To give readers a sense of the time required to cover a specific unit, this report follows the same pattern used in other curricula reports. The CE2016 steering committee has chosen to express time in hours, specifically in core hours. This corresponds to the in-class time required to present the material within a knowledge unit in a traditional lecture-oriented format. Hence, one "core hour" or one lecture hour is one 50-minute period.

To dispel any potential confusion, however, it is important to underscore the following observations about the use of lecture hours as a measure.

- This report does not seek to endorse the lecture format. Even though we have used a metric with its roots in a classical, lecture-oriented form, we believe that other styles can be at least as effective, particularly given recent improvements in educational technology. These include forms such as flipped classrooms, massive open online courses (MOOCs), blended learning, pre-recorded lectures, and seminars. For some of these styles, the notion of hours may be difficult to apply. Even so, the time specifications serve as a comparative metric, in the sense that five core hours will presumably take approximately five times as much time or effort to address as one core-hour, independent of the teaching style.
- The hours specified for a laboratory component to a curriculum often have three contact laboratory hours equivalent to one lecture hour. That is, 150 minutes (three 50-minute contact lab hours) is equivalent to one lecture hour. This calculation varies from institution to institution. Notwithstanding, since laboratories are necessary for a computer engineering program, it is important to include laboratory hours in calculating the time needed to cover a knowledge unit.
- The hours specified do not include time spent outside of class. The time assigned to a knowledge 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 is approximately three times the in-class time. Thus, a unit listed as requiring three hours typically entails a total of twelve hours (three in class and nine outside class).
- The hours listed for a knowledge unit represent a minimum level of coverage. Users should interpret the time measurements we have assigned for each knowledge unit as a minimal amount of time necessary to enable a student to achieve related learning outcomes for that unit. Many instructors will find that delivery of material to the level of depth that they wish to incorporate will take much longer than this; it is always appropriate to spend more time on a unit than the recommended minimum.

3.1.3 Tags for KAs and KUs

We identify a knowledge area with a tag such as CE-NWK, representing the "Computer Networks" knowledge area for computer engineering. We identify each knowledge unit by adding a numeric suffix to the area identification—as an example, CE-NWK-2 is the second knowledge unit within the computer network knowledge area. Supplementary knowledge units have only elective learning outcomes and do not contain any recommended core hours.

3.1.4 Common KUs

Within each knowledge area, the first KU is "History and overview" and the second is "Relevant tools, standards and/or engineering constraints." These KUs provide context for the rest of the KA. The brief history and overview provides context for the learning outcomes, including important contributors to, and developments in the area.

Engineering practice requires the use of modern tools and contemporary standards, which will change over time. The extent of these KUs vary greatly by knowledge area and the goals of the program.

3.2 Learning Outcomes

To capture the sense of what students should learn from each knowledge unit, this report associates *learning outcomes* with each knowledge unit. The emphasis on *learning* is important. Taxonomies of verbs such as "define" or "evaluate" are useful to describe the expected depth of learning. Levels of learning range from basic abilities, such as reciting definitions, to advanced abilities, such as engaging in synthesis and evaluation. The verbs used to describe learning outcomes in KUs were influenced by Bloom's taxonomy [Bloom 1956]. Hence, *learning outcomes* provide a mechanism for describing not just knowledge and relevant practical skills, but also personal and transferable skills. They describe what we expect a student will know or can do by the time of graduation. The minimal desired depth of coverage associated with each knowledge unit can be inferred from the language used to express the learning outcomes. Learning outcomes are not limited to knowledge units, and may be associated with a class activity, a course, or even a degree program.

In this report, the steering committee has tried to limit the number of learning outcomes to emphasize essential skills and knowledge. Programs may choose to structure the curriculum so that students demonstrate their attainment of knowledge and skills in a wide variety of ways. Imaginative approaches to assessment of learning outcomes can lead to unique expressions of a range of skills in well-conceived assignments.

3.3 Summary of the CE body of knowledge

Table 3.1 lists the twelve knowledge areas that form the CE body of knowledge. Table 3.2 shows the twelve CE knowledge areas as presented in this report together with their associated knowledge units. This is the CE body of knowledge. Table 3.2 also shows the core hours (core lecture hours) associated with each area and each unit. For example, *CE-ESY-5 Parallel input and output [3]* in Table 3.2 indicates that "parallel input and output" should have a relative emphasis measured by three core lecture hours and it belongs to the fifth knowledge unit of the "embedded systems" knowledge area, which is core for a computer engineering degree program. The absence of a number such as [2] means the KU is not core; therefore, it is supplementary. Note that the CE2016 steering committee has chosen to approximate these KA hours to the nearest "fives" value to provide another dimension of flexibility for evolving computer engineering programs. Appendix A shows the contents of the knowledge areas and their associated knowledge units.

Table 3.1: CE2016 Knowledge Areas

CE-CAE	Circuits and Electronics	CE-PPP	Preparation for Professional Practice
CE-CAL	Computing Algorithms	CE-SEC	Information Security
CE-CAO	Computer Architecture and Organization	CE-SGP	Signal Processing
CE-DIG	Digital Design	CE-SPE	Systems and Project Engineering
CE-ESY	Embedded Systems	CE-SRM	Systems Resource Management
CE-NWK	Computer Networks	CE-SWD	Software Design

Table 3.2: CE2016 Body of Knowledge
(CE Core Hours: 420)

Knowledge Areas and Knowledge Units	
CE-CAE	**Circuits and Electronics** [50 core hours]
CE-CAE-1	History and overview [1]
CE-CAE-2	Relevant tools, standards, and/or engineering constraints [3]
CE-CAE-3	Electrical quantities and basic elements [4]
CE-CAE-4	Electrical circuits [11]
CE-CAE-5	Electronic materials, diodes, and bipolar transistors [7]
CE-CAE-6	MOS transistor circuits, timing, and power [12]
CE-CAE-7	Storage cell architecture [3]
CE-CAE-8	Interfacing logic families [3]
CE-CAE-9	Operational amplifiers [3]
CE-CAE-10	Mixed-signal circuit design [3]
CE-CAE-11	Design parameters and issues
CE-CAE-12	Circuit modeling and simulation methods
CE-CAL	**Computing Algorithms** [30 core hours]
CE-CAL-1	History and overview [1]
CE-CAL-2	Relevant tools, standards and/or engineering constraints [1]
CE-CAL-3	Basic algorithmic analysis [4]
CE-CAL-4	Algorithmic strategies [6]
CE-CAL-5	Classic algorithms for common tasks [3]
CE-CAL-6	Analysis and design of application-specific algorithms [6]
CE-CAL-7	Parallel algorithms and multi-threading [6]
CE-CAL-8	Algorithmic complexity [3]
CE-CAL-9	Scheduling algorithms
CE-CAL-10	Basic computability theory
CE-CAO	**Computer Architecture and Organization** [60 core hours]
CE-CAO-1	History and overview [1]
CE-CAO-2	Relevant tools, standards and/or engineering constraints [1]
CE-CAO-3	Instruction set architecture [10]
CE-CAO-4	Measuring performance [3]
CE-CAO-5	Computer arithmetic [3]
CE-CAO-6	Processor organization [10]
CE-CAO-7	Memory system organization and architectures [9]
CE-CAO-8	Input/Output interfacing and communication [7]
CE-CAO-9	Peripheral subsystems [7]
CE-CAO-10	Multi/Many-core architectures [5]
CE-CAO-11	Distributed system architectures [4]
CE-DIG	**Digital Design** [50 core hours]
CE-DIG-1	History and overview [1]
CE-DIG-2	Relevant tools, standards, and/or engineering constraints [2]
CE-DIG-3	Number systems and data encoding [3]
CE-DIG-4	Boolean algebra applications [3]
CE-DIG-5	Basic logic circuits [6]
CE-DIG-6	Modular design of combinational circuits [8]
CE-DIG-7	Modular design of sequential circuits [9]
CE-DIG-8	Control and datapath design [9]
CE-DIG-9	Design with programmable logic [4]
CE-DIG-10	System design constraints [5]
CE-DIG-11	Fault models, testing, and design for testability
CE-ESY	**Embedded Systems** [40 core hours]
CE-ESY-1	History and overview [1]
CE-ESY-2	Relevant tools, standards, and/or engineering constraints [2]
CE-ESY-3	Characteristics of embedded systems [2]
CE-ESY-4	Basic software techniques for embedded applications [3]
CE-ESY-5	Parallel input and output [3]
CE-ESY-6	Asynchronous and synchronous serial communication [6]
CE-ESY-7	Periodic interrupts, waveform generation, time measurement [3]
CE-ESY-8	Data acquisition, control, sensors, actuators [4]
CE-ESY-9	Implementation strategies for complex embedded systems [7]
CE-ESY-10	Techniques for low-power operation [3]
CE-ESY-11	Mobile and networked embedded systems [3]
CE-ESY-12	Advanced input/output issues [3]
CE-ESY-13	Computing platforms for embedded systems
CE-NWK	**Computer Networks** [20 core hours]
CE-NWK-1	History and overview [1]
CE-NWK-2	Relevant tools, standards, and/or engineering constraints [1]
CE-NWK-3	Network architecture [4]
CE-NWK-4	Local and wide area networks [4]
CE-NWK-5	Wireless and mobile networks [2]
CE-NWK-6	Network protocols [3]
CE-NWK-7	Network applications [2]
CE-NWK-8	Network management [3]
CE-NWK-9	Data communications
CE-NWK-10	Performance evaluation
CE-NWK-11	Wireless sensor networks
CE-PPP	**Preparation for Professional Practice** [20 core hours]
CE-PPP-1	History and overview [1]
CE-PPP-2	Relevant tools, standards, and/or engineering constraints [1]
CE-PPP-3	Effective communication strategies [2]
CE-PPP-4	Interdisciplinary team approaches [1]
CE-PPP-5	Philosophical frameworks and cultural issues [2]
CE-PPP-6	Engineering solutions and societal effects [2]
CE-PPP-7	Professional and ethical responsibilities [3]
CE-PPP-8	Intellectual property and legal issues [3]
CE-PPP-9	Contemporary issues [2]
CE-PPP-10	Business and management issues [3]
CE-PPP-11	Tradeoffs in professional practice
CE-SEC	**Information Security** [20 core hours]
CE-SEC-1	History and overview [2]
CE-SEC-2	Relevant tools, standards, and/or engineering constraints [2]
CE-SEC-3	Data security and integrity [1]
CE-SEC-4	Vulnerabilities: technical and human factors [4]
CE-SEC-5	Resource protection models [1]
CE-SEC-6	Secret and public key cryptography [3]
CE-SEC-7	Message authentication codes [1]
CE-SEC-8	Network and web security [3]
CE-SEC-9	Authentication [1]
CE-SEC-10	Trusted computing [1]
CE-SEC-11	Side-channel attacks [1]

Knowledge Areas and Knowledge Units			
CE-SGP	**Signal Processing** [30 core hours]	**CE-SPE**	**Systems and Project Engineering** [35 core hours]
CE-SGP-1	History and overview [1]	CE-SPE-1	History and overview [1]
CE-SGP-2	Relevant tools, standards, and/or engineering constraints [3]	CE-SPE-2	Relevant tools, standards and/or engineering constraints [3]
CE-SGP-3	Convolution [3]	CE-SPE-3	Project management principles [3]
CE-SGP-4	Transform analysis [5]	CE-SPE-4	User experience* [6]
CE-SGP-5	Frequency response [5]	CE-SPE-5	Risk, dependability, safety and fault tolerance [3]
CE-SGP-6	Sampling and aliasing [3]	CE-SPE-6	Hardware and software processes [3]
CE-SGP-7	Digital spectra and discrete transforms [6]	CE-SPE-7	Requirements analysis and elicitation [2]
CE-SGP-8	Finite and infinite impulse response filter design [4]	CE-SPE-8	System specifications [2]
CE-SGP-9	Window functions	CE-SPE-9	System architectural design and evaluation [4]
CE-SGP-10	Multimedia processing	CE-SPE-10	Concurrent hardware and software design [3]
CE-SGP-11	Control system theory and applications	CE-SPE-11	System integration, testing and validation [3]
		CE-SPE-12	Maintainability, sustainability, manufacturability [2]
CE-SRM	**Systems Resource Management** [20 core hours]	**CE-SWD**	**Software Design** [45 core hours]
CE-SRM-1	History and overview [1]	CE-SWD-1	History and overview [1]
CE-SRM-2	Relevant tools, standards, and/or engineering constraints [1]	CE-SWD-2	Relevant tools, standards, and/or engineering constraints [3]
CE-SRM-3	Managing system resources [8]	CE-SWD-3	Programming constructs and paradigms [12]
CE-SRM-4	Real-time operating system design [4]	CE-SWD-4	Problem-solving strategies [5]
CE-SRM-5	Operating systems for mobile devices [3]	CE-SWD-5	Data structures [5]
CE-SRM-6	Support for concurrent processing [3]	CE-SWD-6	Recursion [3]
CE-SRM-7	System performance evaluation	CE-SWD-7	Object-oriented design [4]
CE-SRM-8	Support for virtualization	CE-SWD-8	Software testing and quality [5]
		CE-SWD-9	Data modeling [2]
		CE-SWD-10	Database systems [3]
		CE-SWD-11	Event-driven and concurrent programming [2]
		CE-SWD-12	Using application programming interfaces
		CE-SWD-13	Data mining
		CE-SWD-14	Data visualization

* User experience (UX) was formerly known as human-computer interaction (HCI)

3.3.1 Related mathematics

Table 3.3 describes the mathematical component of the CE body of knowledge. The CE2016 steering committee recommends that a robust computer engineering program have at least four areas of capability that require at least 120 hours in mathematics to produce a competent CE professional for the 2020s. Clearly, programs typically include much more mathematics to achieve their goals. The four areas include analysis of continuous functions (calculus), discrete structures, linear algebra, and probability and statistics; these four areas emphasize what the steering committee considers essential to computer engineering.

3.3.2 Related science

The CE2016 steering committee has elected not to recommend specific science areas or the number of hours that a program should devote to science. However, it does recommend that students undertaking computer engineering as a program include as much natural science (e.g., biology or chemistry in addition to physics) as appropriate so that they obtain a command of the scientific bases for engineering. The reason for a science recommendation is that students in the engineering field should develop strong analytical thinking skills and learn empirical and experimental ways of learning. See Chapter 6 for a more detailed discussion on science for computer engineering.

Table 3.3: Related CE Mathematics
(120 Core Hours)

Mathematics Knowledge Areas and Units			
CE-ACF	**Analysis of Continuous Functions** [30 core hours]	**CE-DSC**	**Discrete Structures** [30 core hours]
CE-ACF-1	History and overview [1]	CE-DSC-1	History and overview [1]
CE-ACF-2	Relevant tools and engineering applications [1]	CE-DSC-2	Relevant tools and engineering applications [1]
CE-ACF-3	Differentiation methods [4]	CE-DSC-3	Functions, relations, and sets [6]
CE-ACF-4	Integration methods [6]	CE-DSC-4	Boolean algebra principles [4]
CE-ACF-5	Linear differential equations [8]	CE-DSC-5	First-order logic [6]
CE-ACF-6	Non-linear differential equations [3]	CE-DSC-6	Proof techniques [6]
CE-ACF-7	Partial differential equations [5]	CE-DSC-7	Basics of counting [2]
CE-ACF-8	Functional series [2]	CE-DSC-8	Graph and tree representations and properties [2]
		CE-DSC-9	Iteration and recursion [2]
CE-LAL	**Linear Algebra** [30 core hours]	**CE-PRS**	**Probability and Statistics** [30 core hours]
CE-LAL-1	History and overview [1]	CE-PRS-1	History and overview [1]
CE-LAL-2	Relevant tools and engineering applications [2]	CE-PRS-2	Relevant tools and engineering applications [2]
CE-LAL-3	Bases, vector spaces, and orthogonality [4]	CE-PRS-3	Discrete probability [5]
CE-LAL-4	Matrix representations of linear systems [4]	CE-PRS-4	Continuous probability [4]
CE-LAL-5	Matrix inversion [2]	CE-PRS-5	Expectation and deviation [2]
CE-LAL-6	Linear transformations [3]	CE-PRS-6	Stochastic processes [4]
CE-LAL-7	Solution of linear systems [3]	CE-PRS-7	Sampling distributions [4]
CE-LAL-8	Numerical solution of non-linear systems [4]	CE-PRS-8	Estimation [4]
CE-LAL-9	System transformations [3]	CE-PRS-9	Hypothesis tests [2]
CE-LAL-10	Eigensystems [4]	CE-PRS-10	Correlation and regression [2]

3.3.3 The role of software

While all computer engineers need some familiarity with software development and implementation, the BoK does not specify specific programming languages or operating systems. The learning outcomes in the BoK emphasize higher-level design concepts and interactions between hardware and software. Some computer engineers may develop operating systems, compilers, and other software tools; however, the primary focus of the discipline is on their use in designing systems to meet specified needs. Depending on its goals and the preparation of its students, an academic program may include additional fundamental software knowledge and skills beyond those listed in the CE BoK (e.g., operating system design) to prepare a competent computer engineer.

3.4 CE2016 BoK compared with CE2004 BoK

The CE2004 report recommended a body of knowledge that contained 420 core hours of computer engineering plus 66 hours in mathematics totaling 486 core hours. This CE2016 report recommends a body of knowledge that contains 420 core hours of computer engineering plus 120 hours in mathematics totaling 540 core hours. Table 3.4 illustrates a comparison between the CE2016 and the CE2004 knowledge areas with corresponding hours.

Table 3.4 shows that the body of knowledge for CE2016 has 12 knowledge areas while the body of knowledge for CE2004 had 16 knowledge areas. The situation might give the impression that the BoK for CE2016 reflects a reduction in content. This is not the case. The CE2016 steering committee has merged five of the KAs in CE2004 into the CE2016 KAs while it minimized some areas such as VLSI design in favor of expanded emphasis for system on chip (SOC) technologies and embedded systems that are more reflective of directions in which the discipline has grown. The knowledge units for CE2016 shown in Table 3.2 and in Appendix A show this contemporary approach.

Table 3.4: Body of Knowledge: CE2016 vs. CE2004 Knowledge Areas

CE2016 Knowledge Areas		Core Hours	CE2004 Knowledge Areas		Core Hours
CE-CAE	Circuits and Electronics	50	CE-CSG	Circuits and Signals	43
CE-CAL	Computing Algorithms	30	CE-ALG	Algorithms	30
CE-CAO	Computer Architecture & Organization	60	CE-CAO	Computer Architecture & Organization	63
CE-SPE	Systems and Project Engineering	35	CE-CSE	Computer Systems Engineering	18
CE-DIG	Digital Design	50	CE-DIG	Digital Logic	57
CE-ESY	Embedded Systems	40	CE-ESY	Embedded Systems	20
CE-NWK	Computer Networks	20	CE-NWK	Computer Networks	21
CE-PPP	Preparation for Professional Practice	20	CE-SPR	Social and Professional Issues	16
CE-SEC	Information Security	20			
CE-SGP	Signal Processing	30	CE-DSP	Digital Signal Processing	17
CE-SRM	Systems Resource Management	20	CE-OPS	Operating Systems	20
CE-SWD	Software Design	45	CE-SWE	Software Engineering	13
			CE-DBS	Database Systems	5
			CE-ELE	Electronics	40
			CE-HCI	Human-Computer Interaction	8
			CE-PRF	Programming Fundamentals	39
			CE-VLS	VLSI Design and Fabrication	10
	Total:	420		Total:	420

3.5 Rationale for number of core hours in computer engineering

As mentioned, the CE2016 body of knowledge contains 420 core hours. Additionally, the CE2016 mathematics recommendation contains 120 core hours. As defined in section 3.1.2, one core hour represents the equivalent of one, 50-minute lecture hour.

Within a typical semester system, a four-year undergraduate program typically has at least 120 semester credit hours. One year of study consists of two semesters or 30 semester hours and one semester consists of 15 semester hours (or credits). We define one credit (or one semester hour) as the equivalent of one lecture hour (one 50-minute lecture) per week for 14 weeks plus a week for examinations. Hence, thirty semester hours (one year of study) is equivalent to 30×14 = 420 lecture hours.

The CE2016 steering committee believes that 420 core hours reflects a *minimal* curricular component needed to produce a competent computer engineer with the understanding that most programs will choose to include more hours devoted to computer engineering content. Additionally, a student in computer engineering should experience at least the equivalent of 120 lecture hours in mathematics. We acknowledge that 120 core hours of mathematics is insufficient to produce a competent computer engineer, so the knowledge units in the mathematics component focus only on the knowledge and skills that support essential components of the computer engineering content areas. It is up to each program to decide the quantity and level of mathematics for its graduates. See Chapter 6 for more discussion on this issue.

3.6 Curricular models

The discussion in the previous section provides a minimal framework for generating a body of knowledge for a computer engineering program. Strong CE programs would ordinarily include much more mathematics, science, and CE content to produce competent and competitive graduates for computing industries or for graduate studies. While the 420 core hours within the body of knowledge encompass the learning outcomes that computer engineering students should achieve, not all computer engineering programs will be able to include the entire set.

This may be the result of institutional or programmatic goals and/or availability of resources. The expectation, however, is that strong computer engineering programs would achieve all learning outcomes.

A three-year model includes one-half year of mathematics and science, one year of computer engineering core, one year of computer engineering supplementary, one-half year of general and additional engineering studies. This model is useful in those locations where general studies, mathematics, and science precede university studies. A four-year model program includes one year of mathematics and science, one year of computer engineering core, one year of computer engineering supplementary, one year of general and additional engineering studies. The model is adaptable to many worldwide systems of study. Discussion and samples of curricular representations appear in Chapter 6 and in Appendix B.

Chapter 4

Engineering Practice and the Computer Engineering Curriculum

By its very nature, a computer engineering curriculum 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 (specifically, computer engineering), teaching them to think and function as engineers, and setting expectations for the future. Preparation for practice is essential because many graduates from four-year programs directly begin professional practice in much of the world.

The field of computer engineering (CE) has developed, matured, and expanded throughout industry. CE professionals apply their skills in a broad range of diverse career sectors that include business, industry, government, services, organizations, and other structured entities that use computers to automate or drive their products or services efficiently.

People seeking CE careers have a great potential for success. A recent study by the Bureau of Labor Statistics (BLS) shows that computer occupations are expected to increase 17.7% by 2022, with information security leading by 36.5% [BLS]. Unfortunately, although jobs are and will be available, finding qualified people to fill them is often difficult. Students graduating from engineering programs such as computer engineering often do not have all the attributes to fill the needs of industry. They may have technical skills acquired from their studies, but they lack other skills needed “to fit” within an industry or government environment.

Students who graduate from a four-year university program assume that the baccalaureate degree is a sufficient qualification to attain a position. This understanding may be true in some fields, but not in computer engineering. Belief in this myth has stymied many a job hunter worldwide. The degree credential is likely to be necessary, but it is not sufficient for a position. A general understanding exists in engineering and other fields that a successful professional must be a good communicator, a strong team player, and a person with passion to succeed. Hence, having a degree is not sufficient to secure employment.

Some people even believe that a graduate of a CE program who has a high grade-point-average (GPA) is more likely to attain a position than one who has a lower GPA. This is another mythical belief. A graduate having a high GPA is commendable. However, if s/he does not have the passion and drive, does not work well with others, does not communicate effectively, and does not have the requisite personal skills, chances are that the person will not pass the first interview.

The role of this chapter is to go beyond the body of knowledge introduced in Chapter 3 and examine the basic skills necessary to enable the computer engineering graduate to apply this body of knowledge to real-world problems and situations. Chapter 5 will then address the important matter of professionalism, and Chapter 6 will consider overall curriculum design, along with introducing sample curricula implementations available in Appendix B.

4.1 The nature of computer engineering

An important initial aspect of the engineering ethos relates to acquiring the background necessary to understand and 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 fulfills this role.

This basic material then provides an underpinning for additional material whose ultimate expression is the building of better or novel computer-based 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.

4.2 Strategies for Emerging Technologies

The fields of computing and engineering have changed rapidly in recent times and there is an unwritten promise that the change in these areas will accelerate drastically in the future. Hence, computer engineers must have the background to adapt to new and emerging technologies in an agile manner. They should be able to identify contributors to emerging technologies and identify companies that have failed because they did not adapt to a changing field. The computer engineer, thus, should at least be aware of the positive and negative consequences of developing emerging technologies.

So, how should computer engineers relate to an era of emerging technologies? One way is to identify stakeholders associated with some of these technologies and to identify some strategic assumptions and social values related to the development and application of these new areas. Often, industry breaks scientific barriers to formulate such strategies; sometimes governments set strategic polices to expand or confine these strategies. Standards might even emerge in dealing with emerging technologies. These strategies could involve applied technologies; others could be conceptual in nature.

4.2.1 Applied Emerging Technologies

Computer engineers should be aware of applied emerging technologies. These technologies already exist in the market place but they are sufficiently new that their influence on society is not completely known. Students should be able to identify some applied emerging technologies and indicate their effects on computer engineering. The identification and effects of applied emerging technologies on computer engineering is useful in producing a competent engineer who can contribute to the profession in a changing world.

The computer engineering curriculum should allow the exploration of applied emerging technologies. For example, teachers might encourage examination of ways in which 3D printers might produce artifacts that are harmful to society or describe the challenges one would face in designing and producing integrated circuits. As another example, students should be able to explain ways in which nanotechnology or the internet of things (IoT) can transform the technological workplace. Computer engineers are already or will soon be interacting with optical, biological, or quantum computers or they will be designing a new-age robotic system for manufacturing. These newly emergent and modern technologies present challenges to computer engineering students and practitioners that could involve financial and ethical tradeoffs affecting professional practice in a changing world.

4.2.2 Conceptual Emerging Technologies

Computer engineers should also be aware of conceptual emerging technologies. These technologies are those that exist in some developing state with recent entrance or possible entrance in the market place. Students should be able to identify some conceptual emerging technologies and indicate some of their effects on computer engineering.

The computer engineering curriculum should allow exploration of new inventions that have yet to emerge as viable technologies. For example, teachers might encourage exploration of ways in which a computer engineer would design environments involving augmented reality and virtual worlds or ways in which big data and data analytics might affect the work of a computer engineer. These activities would affect design constraints needed to

address emerging areas such as computational biology and bioinformatics. Additionally, it would be useful to have students explore the role of a computer engineer in an era of machine learning and intelligent systems, or discuss engineering strategies needed in developing a culture of green computing and sustainability. New technologies might even expose safety issues affecting the field of computer engineering. Awareness of these and other issues are important in developing a well-rounded and social conscious computer engineer.

4.3 Design in the curriculum

In Chapter 2, a brief discussion of the characteristics of a computer engineer included the ability of the engineer to design and provide, as well, a definition of engineering design. The following sections provide guidance on how design can be incorporated within the computer engineering curriculum.

4.3.1 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 for deciding which choices one must make. Depending on the specific application requirements, the design context might 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 accumulated technical knowledge and prior design experiences of the students.

One area of concern to the computer engineer is the software/hardware interface, where difficult trade-off decisions often provide engineering challenges. Considerations on this interface or 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 they lead 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 that the system operates properly and reliably.

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

4.3.2 The culminating design experience

The concept of a culminating design project is widely valued as an important experience occurring toward the end of a curriculum. Students consider a significant problem associated with a discipline and, in solving the problem, demonstrate their ability to apply methodically engineering principles to generate a solution. For computer engineering, the solution typically involves the design and implementation of a system or subsystem containing both hardware and software components and considering a variety of interactions and tradeoffs. The design experience is often team-based, which best reflects industry practice. Ideally, the design experience should incorporate engineering standards and realistic constraints to represent what might 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 the following:

- engagement at the level of the structure and function of the overall system, including consideration of relevant ethical and social implications
- demonstration of the ability to integrate concepts from several different subjects into a complete system that provides a solution
- demonstration of the application of tools and principles associated with computer engineering
- production of a well-written document (and other materials) detailing the design and the design experience
- demonstration of creativity and innovation
- development of time management and planning skills
- self-awareness opportunities provided by a measurement of achievement

A program could find evidence that students have attained specific knowledge and skills in many ways. These might include a demonstration, a presentation, an oral examination, production of a web site, or industry review. Although not listed in the core body of knowledge, the culminating design experience must be an integral part of the undergraduate experience. While many core learning outcomes will occur before the culminating design experience, some may best be introduced during it, thereby emphasizing the need for lifelong learning

4.4 Laboratory experiences

Laboratory experiences are an essential part of the computer engineering curriculum and they serve 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 using that data to correct or improve the design and to verify that it meets specifications.

Introductory laboratories (e.g., programming laboratories) are typically somewhat directed and designed to reinforce concepts presented in lecture classes and homework. Such activities demonstrate specific phenomena or behaviors and they provide experiences such as varying parameters to attain desired characteristics. Intermediate and advanced laboratories should include problems that are more open-ended, requiring students to design and implement solutions or requiring them to design experiments to acquire data needed to complete the design or to measure various characteristics.

It is not the purpose of this report to recommend which is the best method of engaging students in laboratory experiences. Individual programs should seek out options that best suit their needs based on available space, the course objectives, and the resources available.

4.4.1 Computer engineering laboratories

Many courses in computer engineering should contain laboratory experiences. Typically, a laboratory experience lasts two or three hours and it occurs in a location configured with specialized equipment. The depth and breadth of these experiences will vary among institutions. The variation often depends on the time allocated within the curriculum, physical space, and resources. Table 4.1 illustrates the steering committee recommendation on the types of laboratories students studying computer engineering should experience. Some laboratories involve practices that all computer engineering students must or should do. Those laboratories indicated by "●●●●" refer to distinguishing experiences every computer engineer *must* have while those indicated by "●●" refer to experiences every computer engineer *should* have. Availability of supplemental laboratories will depend on local program needs, objectives, and resources; Table 4.1 illustrates some common types of laboratories. Appendix C suggests typical descriptions, configurations, and offerings for laboratories other than Senior Project Design Laboratory, which should reflect the specific needs for individual projects and the mission of each program.

Table 4.1: Types of computer engineering laboratories

Laboratory Type	Must	Should	Supplemental
Circuits and Electronics	●●●●		
Computer Architecture Design			●
Digital Signal Processing			●
Digital Logic and System Design	●●●●		
Embedded Systems	●●●●		
Introduction to Engineering			●
Networking		●●	
Software Design		●●	
Senior Project Design	●●●●		

Table 4.2 shows additional laboratory options that programs could consider.

Table 4.2: Suggested Additional Computer Engineering Laboratories

Audio Engineering	Microwave Measurements
Computers in Manufacturing	Operating Systems
Electrical Energy Systems	Robotics
Graphics	Specialized Electronics Lab
Mechatronics	Teaching Enhancement
	Telecommunications

Laboratories should include some physical implementation of designs such as electronic and digital circuits, bread-boarding, FPGAs/CPLDs, microcontroller-based systems, prototyping, and implementation of firmware. Laboratories should also include application and simulation software to design computer systems including digital systems. Simulation tools present intrinsic value as part of professional computer engineering practice. They are useful in modeling real systems and they are often desirable and necessary to allow students to study systems that are impractical to design and implement given available time and resources. In addition, configuration management and version control software should be used for both hardware and software.

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

- safety in all laboratories, especially where electronic equipment and electricity pose dangers
- proper use of computers and test equipment
- building electrical and electronic circuits and systems
- building and testing software
- understanding processes and issues associated with product development and manufacturing
- recognizing opportunities for trade-offs (such as hardware, software, off-the-shelf IP, time/space, power, modularity, modifiability, security, and efficiency) and being able to make informed decisions in this area

At the formative stages of their education, students often are motivated by the hands-on nature of engineering. Laboratory experiences capitalize 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. Laboratory experiences should help students develop the expertise needed to build new products and to appreciate the important role of technical staff, workshop teams, and professionals from other disciplines.

4.4.2 Software considerations

Software tools and packages related to computer engineering will vary based on the philosophy and needs of each program. Table 4.3 suggests some software that could appear on all machines within specific laboratory settings. Products mentioned in this table are included for illustrative purposes only and represent options available at the time of writing of this report; no endorsement of a specific product is implied. Additionally, it is not envisioned that any program will incorporate every one of these software applications. Each program should determine its own needs and consider including the most current version of appropriate applications.

Table 4.3: Suggestions for possible software applications

Design modeling and simulation • Circuit-level (e.g., SPICE) • Gate-level (schematic entry) • Digital systems (e.g., VHDL, Verilog) • Analog/mixed-signal circuits (e.g., VHDL-AMS, Verilog-AMS) • System-level design (e.g., System Verilog, System C) **Digital hardware prototyping** • FPGA/CPLD development suite • Design file entry and management • Component/IP library support • Device programming • Interactive debugging **Microcontroller system design** • Integrated development environment (IDE) ▪ Design entry/management ▪ Library support • Compilers, assemblers, linkers • Processor simulators/emulators • Device programming • In-circuit test/debug **Mathematics packages** • Problem solving • Data analysis • Modeling and simulation	**Software development** • Integrated development environment (IDE) ▪ Design entry and management ▪ Library support • Compilers (e.g., C, C++, C#, Java, Python) • Operating system support • Source-level debugging • Platform support (e.g., smartphone, tablet) **Integrated circuit/ASIC design** • Design capture and simulation • Synthesis • Physical layout ▪ graphical layout editor ▪ automated layout • Design verification (design rules, layout vs. schematic, parameter extraction) • Design for testability, automatic test pattern generation **Printed circuit board (PCB) design** **Computer-aided design and modeling (CAD tools)** **Laboratory automation and instrumentation** (e.g., IntuiLink or LabVIEW software)	**General computing/productivity** • Web browser • Email • Office suite • PDF reader/editor • Illustration/photo viewer/editor • Multimedia players/editors • File compression/ decompression • File transfer (FTP, SCP, SFTP) • Terminal emulator, remote login, secure shell, X Window client **System engineering tools** • Project management (e.g., GANTT or PERT charts) • Requirements and specifications management (e.g., UML tools) **Other tools** • Robotics software development • Semiconductor device and process modeling (e.g., TCAD) • Microwave, RF and other high-speed/high-frequency design • Electromagnetic field simulation • MEMS design

4.4.3 Open-ended laboratories

The computer engineering curriculum often contains open-ended experiences where true research and development takes place. One might view this as the “ultimate lab experience.” A culminating or capstone design experience usually embodies this open-ended flavor. In such situations, an instructor and a team of students decide on an exploration area and, once decided, the student team begins the research and design process. Programs usually provide a dedicated space where teams can meet and work. These spaces generally contain modern facilities and provide sufficient space for electronic devices (e.g., robots) and other equipment required by the project at hand.

4.4.4 Embedded laboratories

The traditional laboratory experience normally takes place in a room separate from the lecture and at a different time. For example, a graphics course might have lectures on Tuesday and Thursday, with a three-hour laboratory on Wednesday. It is increasingly common in modern educational settings to have laboratories embedded within ordinary courses. One such practice is to have lectures take place within the laboratory itself. Another is to partition a room in some manner and have the lecture take place in one part of the room with the laboratory and its associated equipment in another part of the same room. There is also an emerging trend to have "flipped classrooms" in which lectures are recorded for students to view in advance. Then class time is used to engage the students in active learning, which could include students bringing their own breadboards and instrumentation, and engaging in laboratory exercise.

4.4.5 Technical support

A program needs to have ready access to technical support to develop a "working lab" environment. For example, to create a new antenna design for experimentation, it might be necessary to use existing facilities available at a technical workspace that contains a lathe and welding tools. The center might contain 3-D printers, a wide variety of standard circuit components and chips, and special tools and equipment such as modern microcontroller and/or FPGA development boards for general use.

4.4.6 Student purchases

Another trend is to have students purchase their own laboratory equipment or obtain special free or low-cost versions of software tools. With the cost of some equipment at remarkably low prices and the size of special packages reduced to a simple toolkit, students can now purchase their own software and equipment and bring it to their laboratory classes. (Departments usually have a limited number of spares available for student checkout if needed.) This has the advantage of allowing students to conduct laboratory experiments outside the formal laboratory environment. Examples of student purchases include:

- personal instrumentation (e.g., oscilloscope, multimeter, logic analyzer, waveform generator) connected to a laptop computer that hosts soft instrument front panels
- breadboard, with integrated power supply, for circuit construction
- low-cost microcontroller and FPGA development boards
- student versions of modeling, simulation, and software development tools

4.5 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. In the rapidly changing world of computer engineering, there are also opportunities for identifying roles for new tools and their applications. 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; tools for modeling and simulating circuits and systems; hardware description languages; tools to automate various design steps, including synthesis, test pattern generation, and physical implementation (e.g., FPGA map-place-route); emulators; and debugging tools. Other hardware design tools include those to support circuit design, printed

circuit design layout, circuit behavior analysis, block diagram creation and editing, communications systems modeling, mixed analog and digital simulation modeling, design rule checking, and virtual instruments.

Software design and analysis tools include operating systems, version control systems, IDEs, GUI designers, language processors, interactive debuggers, and computer-aided software engineering (CASE) tools. General support tools include algorithm development and mathematical analysis environments (such as MATLAB and Mathcad), office software (word processors, spreadsheets, browsers, and search engines), databases, communications software, and project management tools.

Not every computer engineering program will incorporate every one of these tools. The program should incorporate appropriate tools throughout the program of study, consistent with the program's goals and objectives. Understanding the limitation of tools and 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 items. Such activities need to be guided by concerns for quality in its many guises—safety, usability, reliability, security, and so on.

4.6 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 students' thinking and ambition
- providing justification for the inclusion and prominence of certain material
- demonstrating the application of theoretical ideas

A program can achieve these attributes through a range of possible routes. These include the use of up-to-date and topical case studies, guided reading, plant visits, speakers from industry, and other diverse paths. This experience can occur at a range of levels, including chip design, development of software tools, and development of 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 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 can 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 the discipline. Some institutions offer cooperative education programs during 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 might work with professionals from other disciplines, such as computer scientists, electrical engineers, mechanical engineers, entrepreneurs, financial experts, marketers, and product designers.

4.7 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 prepared to move into employment with skills and expectations that benefit their employers.

One aspect of this is ensuring that students possess a set of transferable or personal skills such as communication, teamwork, and presentation skills. Transferable skills are those skills a person can use in any occupation and can convey 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 literacy,

information management, career development, self-awareness, and keeping up-to-date with innovations in the field. From a motivational perspective, students should receive formative feedback on these skills in the context of computer engineering and in a way 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 testing is required to ensure that there is not imbalance in the curriculum.

4.7.1 Communication skills

Computer engineers must be able to communicate effectively with diverse audiences. 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 specific aspect of the activity of a computer engineer is to convey and negotiate project requirements and implementation details to a workshop or technical support staff, which in an industrial setting might be local or remote. Providing clear and succinct documentation 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, including technical writing experiences (e.g., specifications, requirements, safety cases, documentation) as well as report writing, including the use of figures, diagrams, and appropriate references;
- develop persuasive presentation materials and make effective oral presentations, both formally and informally;
- understand and offer constructive critiques of the writings and 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.

Although institutions might adopt different strategies to accomplish these goals, each computer engineering student's program 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. These have a central role to play in the life of the engineer. Apart from the obvious need to address areas such as email,

electronic calendars, and code and document repositories, students should engage, at some level, with effective cooperative working and group learning.

4.7.2 Teamwork skills

Few computer engineering professionals can expect to work in isolation for very much of the time. Groups of people working together as a team implement major computer engineering projects. And often, the teams are interdisciplinary in nature. Computer engineering students therefore need to learn about the formation of effective teams and 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 may extend over a full semester or more.

Many of the problems of teamwork relate to communication skills. Where interdisciplinary 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 for reaching consensus and 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 that students acquire these skills as undergraduates, such as with

- opportunities to work in teams beginning relatively early in the curriculum and/or
- a significant project involving the complex design and implementation of some product or prototype, undertaken by a small student team

4.7.3 Soft or personal skills

Industry managers almost unanimously agree that soft skills (or personal skills) are a primary criterion for hiring a graduate in a computing-related position. Conventional wisdom among industry managers dictates that soft skills and technical skills have equal or similar value.

So, what exactly are soft skills? One definition states that soft skills are “desirable qualities for certain forms of employment that do not depend on acquired knowledge: they include common sense, the ability to deal with people, and a positive flexible attitude.” [DICT] Another definition indicates that soft skills are “the character traits and interpersonal skills that characterize a person's relationships with other people.” [INVEST] This definition continues, “soft skills have more to do with who we are than what we know. As such, soft skills encompass the character traits that decide how well one interacts with others, and are usually a definite part of one's personality.” [INVEST]

In the field of computer engineering and in other fields, soft skills often complement hard skills, which are specific learned abilities. Often, we refer to these soft skills as part of social intelligence or “the ability to connect to others in a deep and direct way, to sense and stimulate reactions and desired interactions.” [CHIN] This ability to connect with co-workers in a convincing manner will be extremely important in the future. In fact, it is likely to become the distinguishing factor between those who are successful in their careers and those who are not.

4.7.4 Experience

Even with the requisite personal, communication, and teamwork skills, technical knowledge may not be sufficient in certain industry environments without prior industry experience. This chicken-and-egg scenario has haunted university graduates for decades and centuries.

In recent years, the engineering industry has been fortunate to have many opportunities for part-time or even full-time employment on a temporary basis. Such opportunities can take many forms for students. Engaging in an internship, cooperative-education, or work-study program will not only allow students to gain practical experience, but it might also allow them to gain academic credit and, in many cases, to receive pay for their work. Often, this experience does not allow students to take courses, so the focus is on the practicum and not on passing exams. Any constructive industry experience a student acquires is a definite plus for those seeking permanent employment upon graduation.

Often students work part-time while studying at a university. This blending of real world experience and academic endeavors provides a necessary component to help them decide on future career goals. Academia needs to embrace this experience since this is what helps differentiate one candidate from another. Experience often becomes a key component for success in achieving a position after graduation. Even though the shortage of qualified computing professionals is expected to continue into the 2020s, a complex interviewing scenario with much competition continues to remain for desirable positions [CHIN].

Undergraduate CE programs should explore all possibilities in bridging the experience gap between academia and industry. Developing a robust industry connection should always be a priority. For example, developing a strong professional advisory board is one way to open doors with industry because members of that board will develop a bond with the program. Therefore, academic CE programs should seek all avenues with industry so their graduates have a greater chance of employment and engagement.

4.7.5 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 equipping students with the mechanisms for achieving this.

There are a number of appropriate basic strategies. The curriculum itself must be current, the equipment must be up-to-date, and faculty members must be engaged in relevant scholarship. Reference material such as textbooks, software, web sites, MOOCs, case studies, and illustrations can be part of the learning experience with the aim of identifying sources of current and interesting information. Additionally, students should be familiar with the wide variety of related resources provided by their professional societies, such as the ACM Learning Center and the IEEE Speakers Bureau.

Lifelong learning is essentially an attitude of mind, including recognition of the need for new knowledge, the ability to search for and identify relevant information, and the ability to evaluate and apply that knowledge. Institutions can foster such attitudes with novel approaches to teaching and learning that continually question and challenge situations and by highlighting opportunities for advances. Instructors can challenge students by 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.

4.7.6 Business perspectives

To complement the technical side of their experiences, computer engineers need an understanding of the various nontechnical 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
- understanding various approaches to the protection of intellectual property
- appreciating the benefits of teamwork, often interdisciplinary in nature

In addition, students need to understand and appreciate their fiscal responsibilities to their employers. Time translates to money and the importance of completing jobs on schedule is important. The business world can also present tensions between corporate systems and ethics. Students should be aware of the professional challenges that might await them in government or corporate service. Within the computer engineering curriculum, such areas can 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.

4.8 Becoming professionals

As students prepare for their future career, an important consideration is their ability to transition from an academic environment to a career within a corporation, organization, academic institution, or even an entrepreneurial environment. This transition could be difficult if an individual has not received the proper exposure to both technical and complementary skills during his or her academic career.

Adaptability is a personality trait that is especially important within the computing industry, and will be very important for career success in the future. The Gartner Group is predicting a new “digital industrial revolution” that will force major changes on us—from 3D printing to the use of non-overridable smart systems to wearable computing [ALIBAB].

In addition to focusing on the industry and gaining valuable work experience while attending a college or university, it is important that students nearing graduation are prepared for important interviews. This preparation includes structuring their resumes into a format that highlights their technology background. What distinguishes a technical resume from a standard one is the emphasis on attributes such as specific technical and personal skill sets as well as industry certifications. Being able to handle a successful interview is a career skill that is essential for students to practice and master during their academic studies. If students are unable to handle the rigors of a career interview, their academic GPA and various scholastic achievements will likely fail them in achieving the desire goal of a useful education in computer engineering.

4.9 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 various contexts
- recognizing the importance of developing effective oral and written communication skills
- recognizing the importance of understanding 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 them 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 it
- 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.

4.10 Graduate and continuing professional education

Because of the rapidly changing field of computing, the steering committee strongly encourages graduates from computer engineering programs to pursue advanced studies in graduate school. Since this report focuses on baccalaureate (undergraduate) programs, interested parties should explore what best meets their interests and needs. Depending on the structure of their undergraduate education, many possibilities exist for graduate work. At that level, depth is necessary as well as an element of specialization.

Students embarking on graduate studies have opportunities to obtain a deeper understanding of hardware issues, software issues, models, the interactions between these issues, and related applications. A combination of theory, practice, application, and attitudes (and, at this level, even innovation) will continue to guide the construction of each module or course. As befits a master's degree, the emphasis on (normally individual) project work especially in the final year is often considerable. For students at that level the projects will typically be addressing innovative concepts that are close to the frontiers of new developments or research.

Chapter 5

Professional Practice

As noted in Chapter 2, people generally understand that computer engineering (along with other engineering disciplines) is a profession, not simply a career field, which imposes certain obligations on its practitioners with respect to the social context of professional practice. Hence, professionalism and professional practice are central in the lives of computer engineers and all engineers for that matter.

Computer engineers differ from other computing specialists in their concentration on the design and implementation of computer systems (including both hardware and software) that affect the public. Hence, computer engineers must consider the professional, societal, and ethical context in which they do their work. This context includes issues such as intellectual property rights embodied by copyrights and patents, legal issues including business contracts and employment agreements, security, and privacy issues as they apply to networks and databases, liability issues as applied to hardware and software, and economic issues as they apply to tradeoffs between product quality and profits.

Professionalism also includes ethical issues related to design decisions and it requires attention to 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. This requires an ability to interact with people (soft skills), understanding cultures and mores, working on and with teams, and communicating effectively using a variety of media for a wide range of audiences, business skills to function within enterprise organizations, and individual skills such as creativity, innovation, and leadership. Therefore, as the field of computing continues to change and as professional interaction becomes more global, an unprecedented opportunity exists to make professional practice a seamless part of the curriculum in computer engineering.

We now show ways in which professional practice can be an integral part of computer engineering and we explore various strategies for incorporating professional practice into the CE curriculum. The individual sections review the underlying rationale, current practice in education, support for professional practice from both the private and public sector, techniques for incorporating professional practice into a curriculum, and strategies for assessing the effectiveness of those techniques.

5.1 Overview of professional practice

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, their employer's image, or their practicing colleagues.

5.1.1 Professional practice and the CE curriculum

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.

It is important to incorporate professional practice into the curriculum because graduates of computer engineering programs will face real-world issues in the workplace, issues such as the needs of the public and private sector, the public's demand for higher quality products, the increasing number of computing liability cases, and the need to promote lifelong learning. In most cases, students enter school without a complete knowledge or appreciation for these issues, and this lack of knowledge is a source of frustration both for those who teach these students and for those who hire them. Indeed, as students learn more about professional practice and its underlying issues, they become more interested in their studies and how they can work well with others. Therefore, incorporating professional practice into the curriculum can serve as a catalyst to stimulate and broaden a student's interest in computing.

5.1.2 Professional needs

Both the private and public sectors have a stake in students learning professional practice. They find that students who have experience with the realities of professional work understand the value of interpersonal skills in collaborating with team members and clients, maintain their focus on producing high-quality work, adhere to strong ethical convictions, contribute their time and talents to worthy outside causes, engage in lifelong learning, and participate in improvements in their organizations.

The connections between employers, graduates, and students are becoming ever more important. The growing demand for better, less defect-ridden products has also increased the pressure to incorporate professional practice into the curriculum. Haphazard web-system design techniques are widely recognized as a significant factor in producing web systems with a high number of defects. Thus, clients are demanding “proof” of reliable system processes before they will sign a contract with provider. Students need to understand the value of establishing face-to-face relationships with clients, agreeing to implementable requirements, and producing the highest quality computer systems possible.

5.2 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 importance are issues related to intellectual property. There is considerable interest in free open-source hardware and software that may have legal and ethical implications. An understanding of patent law is important, particularly when the companies for whom they work may have an active patent program or when entrepreneurs develop their own products. It is also necessary to understand copyright law since many entrepreneurs and 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.

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 foster an awareness of the social context in which the computing systems might be 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 and of the social consequences of taking risks, be accountable for the designs they develop, and be aware of the actions they take and the potential consequences of them. 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 and 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, educational institutions should establish safety guidelines for the proper use of machines and equipment. They should also provide guidelines on interpersonal skills between students, students working in groups, and students interacting with technicians in a laboratory setting. And educational 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 function.

5.3 Professionalism and education

Many strategies currently exist for incorporating professional practice into the curriculum. Among the most common characteristics of these strategies are courses that help students strengthen their communication, problem-solving, and technical skills. Computer engineering programs can foster these skills in computing courses. Alternatively, programs may provide experiences outside computer engineering departments such as in a speech class from a communication department or a technical writing class in an English department. Accreditation bodies, however, usually require not only that students acquire these skills through either general education requirements or courses required specifically for computer engineering, but that they apply these skills in their later academic experiences.

The level of coverage assigned to professional practice varies depending on institutional commitment, departmental resources, and faculty interest. With the growing emphasis on professionalism in accreditation settings, it is likely that institutions will strengthen their commitment to teaching professional practice.

5.3.1 Special student experiences

The following list outlines several potential mechanisms for incorporating additional material on professional practice.

- *Culminating Design Courses*: These courses typically form a one- or two-semester sequence during a student's final year. Usually, students must work in teams to design and implement projects. Often, those projects involve consideration of real-world issues including cost, safety, efficiency, and suitability for the intended user. Students could develop projects that may be solely for the class, or the project may also involve other on- or off-campus clients. Although the emphasis of the course is on project work and student presentations, some material on intellectual property rights, copyrights, patents, law, and ethics may be included.
- *Professionalism, Ethics, and Law Courses*: These courses are one semester long and expose students to issues of professional practice, ethical behavior, and computer law, geographical limits of the authority of different country courts. Study areas included may be history of computing, effect of computers on

society, computing careers, legal and ethical responsibilities, international computer law, and the computing profession.

- *Practicum/Internship/Co-op Programs*: These programs are sponsored by the institution (preferably) or department to allow students the opportunity to work in industry full- or part-time before graduation. Having adequate administrative support for such programs is essential to their success. Students typically work during the summers and/or from one to three semesters while they are engaged in their four-year degree. The students who do a co-op or an internship generally do so off campus and so may interrupt their education for a summer or a semester. Students usually receive payment for their work, but may also receive course credit.
- *Team-based Implementation Courses*: These courses emphasize the process of CE system development and typically include a team project. The course could include development processes, project management, economics, risk management, requirements engineering, design, implementation, maintenance, software and hardware retirement, system quality assurance, ethics, and teamwork. The coverage is usually broad rather than in-depth.
- *Entrepreneurial Innovation Courses*: The computing industry needs disruptive innovation and major companies to provide new technologies and more job opportunities. This course discusses the basics every manager needs to organize successful technology-driven innovation in entrepreneurial settings or established firms—innovation that will integrate creativity and design thinking in the business functions of engineering, management, communication, and commerce. The students will evaluate, research, write, and present business plans using their knowledge of the entrepreneurial process.

Many courses outside computer engineering departments can also help students to develop stronger professional skills. Such courses include, but are not limited to, philosophical ethics, psychology, business management, economics, technical communications, and engineering design.

5.3.2 Administration, faculty, and student roles

At the highest institutional level, the administration must support faculty professional and departmental development activities. Such activities may include consulting work, professional society and community service, summer fellowships, obtaining certifications and professional licensure, achieving accreditation, forming industrial advisory boards with appropriate charters, establishing co-op/internship/practicum programs for course credit, and creating more liaisons with the private and public sectors. Such activities can be extremely time-consuming. They are, however, enormously valuable to both the individual and the institution, which must consider these activities in decisions of promotion and tenure.

Faculty and students can work together by jointly adopting, promoting, and enforcing ethical and professional behavior guidelines set by professional societies. Faculty should join professional societies and help to establish student chapters of those societies at their institutions. Through student chapters, faculty can give awards for significant achievement in course work, service to the community, or related professional activities. In addition, student chapters may provide a forum for working with potential employers and be instrumental in obtaining donations, speakers, and mentors from outside the institution.

5.3.3 Incorporating Professional Practice into the Curriculum

The incorporation of professional practice must be a conscious and proactive effort because much of the material blends into the fabric of existing curricula. For example, the introductory courses in the major can include discussion and assignments on the impact of computing and the internet on society and the importance of professional practice. When students undertake their second-year courses, they should start to keep records of their work, just as a professional engineer does in the form of requirements, design, and test documents.

Additional material such as computer history, digital libraries, search techniques, techniques for tackling ill-defined problems, teamwork with individual accountability, real-life ethical issues, standards and guidelines, legal constraints and requirements, and the philosophical basis for ethical arguments may also appear either in a dedicated course or distributed throughout the curriculum. The distributed approach has the advantage of presenting this material in the context of a real application area. On the other hand, the distributed approach can be problematic in that teachers often minimize professional practice in the scramble to find adequate time for the technical material. Projects, however, may provide a natural outlet for much of this material particularly if faculty members can recruit external clients needing non-critical systems. When they engage in service-learning projects in the community or work with external clients, students begin to see the necessity for ethical behavior in very different terms. Thus, those students learn much more about ways to meet the needs of a client's ill-defined problem. However, no matter how teachers integrate professional practice into the curriculum, it is critical that they reinforce this material with exercises, projects, and exams.

For departments with sufficient flexibility in credit hours or with adequate numbers of faculty members and resources, courses dedicated to teaching professional practice may be appropriate. These courses include those in professional practice, ethics, and computer law, as well as senior capstone and other appropriate courses. Additionally, more advanced courses on web system economics, quality, safety, and security may be part of the experience. These courses could come from disciplines outside of computer engineering and they would still have a profound effect on the professional development of students.

The inclusion of professional ethics in a computing engineering curriculum is fundamental to the discipline. A listing of learning outcomes appears under the professional practice (CE-PPP) area as part of the body of knowledge for computer engineering (see Appendix A).

5.3.4 Professionalism and student experiences

Faculty members can promote professionalism by establishing an infrastructure in which student work falls under common standards that encourage its professional completion. The infrastructure may include the following elements:

- reviewing assignments, projects, and exams for appropriate inclusion of professional practice material
- critically reviewing and establishing sound measurements (e.g., outcomes based) on student work to show student progress and improvement
- getting students involved in the review and evaluation process so that they obtain a better sense of the process
- employing professionals in the private and public sectors to help assess student project work
- using standardized tests to measure overall student progress
- conducting post-graduation surveys of alumni to see how they thought their education prepared them for their careers
- obtaining program accreditation to demonstrate compliance with certain education standards for professional practice
- synchronizing course labs with employer needs to ensure students learn job skills required by employers

The review and evaluation process should encourage students to employ good technical practice and high standards of integrity. It should discourage students from attempting to complete work without giving themselves enough time or in a haphazard manner such as starting and barely completing work the night before an assignment is due. The assessment process should hold students accountable on an individual basis even if they work collectively in a team. It should have a consistent set of measurements so students become accustomed to using them and they learn how to associate them with their progress.

5.4 Professionalism and the workplace

Most students graduating from universities go on to employment in the private or public sector. Industry and governments, in their role as the primary consumer of graduating students, play an important role in helping educational institutions promote professional practice. As an example, students who take advantage of industrial co-ops or government internships may mature faster in their problem-solving skills and become more serious about their education. Such internships may also help the industries that offer them, since a student who has an internship with a company may choose to work there after graduation. With private/public sector support, professional practice coverage provides a necessary augmentation both inside and outside the classroom.

5.4.1 Private and public sectors

One of the most important ways that the private and public sectors can support the education process is to encourage their employees to play a greater role in helping to train students. These employees can offer support in several ways. They can

- function in the role of mentors to students working on projects;
- give special presentations to classes telling students and faculty about their firm, their work, and their development processes;
- take part-time positions as adjunct instructors to strengthen a university's course offerings;
- provide in-house training materials and/or classes to faculty and students in specialized research, process, or software tool areas; and
- serve on industrial advisory boards, which service allows them to provide valuable feedback to the department and institution about the strengths and weaknesses of the students.

In each of these ways, institutions in the private and public sectors can establish important lines of communication with the educational institutions that provide them with their future employees.

In addition to the various opportunities that take place on campus, industry and government also contribute to the development of strong professional practice by bringing students and faculty into environments outside of academia. Students and faculty may take field trips to local firms and begin to establish better relationships. Over a longer term, co-op, practicum, and internship opportunities give students a better understanding of what life on the job will be like. In addition, students may become more interested in their studies and use that renewed interest to increase their marketable potential. Students may also form a bond with specific employers and be more likely to return to that firm after graduation. For faculty, consulting opportunities establish a higher level of trust between the faculty member and the company. Because of these initiatives, employers, students, and faculty know more about each other and are more willing to promote each other's welfare.

In what remains one of the most important forms of support, private and public sectors may also make donations or grants to educational institutions and professional societies in the form of hardware, software, product discounts, money, time, and the like. Often, these donations and grants are critical in providing updated resources, such as lab hardware and software, and in funding student scholarships and awards as well as faculty teaching or research awards. They serve to sponsor student programming, design, and educational contests. Grants can enable more research and projects to occur. At this level, private and public sectors help to ensure the viability and progress of future education and advances in the computing field.

Through patience, long-term commitment, understanding of each other's constraints, and learning each other's value systems, institutions in the private or public sector and in education can work together to produce students skilled in professional practice and behaviors. Their cooperative agreement is essential for producing students who value a high ethical standard and the safety of the people who use the products the students will develop as professionals.

5.4.2 Modelling local and international work environments

Just as industry representatives increasingly seek graduates who are "job ready," most students expect to practice computing in the workplace upon graduation without significant additional training. Although the educational experience differs from that of the workplace, educators can ease the transition from academia to the business world by:

- mimicking the computing and networking resources of the work environment
- teaching students how to work in teams
- providing significant project experiences

Introducing these points into the curriculum makes it possible to model significant issues in the local and international work environment. Faculty can discuss and have students apply international, intercultural, and workplace issues within the context of computing resources, teamwork, and projects.

Because computing and networking environments change rapidly and several different ones exist, it is not possible to predict the exact environment that students will use upon graduation. And so, it is not advisable to focus attention in the curriculum on a specific set of tools. Exposure to a wide variety of computing platforms and web system tools provides sound preparation for professional work, resulting in flexible learners rather than students who immaturely cling to their one familiar environment. Naturally, it is also useful to devote considerable attention to a few tools and platforms to develop depth and capability.

As will be discussed in Chapter 6, learning how to work in teams is not a natural process for many students, but it is extremely important. Students should learn to work in both small and large teams so that they acquire planning, budgeting, organizational, and interpersonal skills. Ample course material should support the students in their teamwork. The lecture material may include project scheduling, communication skills, the characteristics of well-functioning and malfunctioning teams, and sources of stress for team environments. Assessment can include the result of a team's work, the individual work of the members, or some combination thereof. Team member behavior may also play a factor in the assessment.

Significant project experiences can enhance the problem-solving skills of students by exposing them to problems that are not well defined or that do not have straightforward solutions. Such projects may be a controlled, in-class experience or have a certain amount of unpredictability that occurs with an outside client. The projects should serve to stretch the student beyond the typical one-person assignments that exercise basic skills in a knowledge area. Beyond that, projects can also cut across several knowledge areas, thereby helping students to bring all their basic skills together.

5.4.3 Certifications

We acknowledge the value of vendor and industry certifications, and we encourage students to pursue them as they see necessary. However, we do not believe that programs should offer academic credit for completion of such certifications or for training exclusively designed to prepare for these certifications unless it also covers relevant learning outcomes defined in this document. We take this position because many institutions offer certification training without having credentials to operate as an institution of higher learning. Most certifications are practice-oriented and they do not focus on underlying theories and concepts. Additionally, many certifications are specific to a given vendor and they have a narrow focus. Therefore, they usually do not meet the learning outcomes necessary for degree in computer engineering.

5.5 Fostering Professionalism

The issues highlighted in the previous sections have led many professional societies to develop committees, professional programs, codes of ethics and professional practice, and other related benefits for their

constituencies. These programs help practitioners understand expected standards of professional conduct and the expectation among member practitioners.

5.5.1 Professional ethical codes

Professional ethics codes provide public information concerning the precepts considered central to the profession. These codes also provide a level playing field for professionals with the prospects of avoiding ethical dilemmas whenever possible. They also help 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.

ACM, IEEE, AITP, and other organizations promote the development of professional responsibility in several ways.

- They develop and promote codes of ethics to which members must adhere. These codes, in general, promote honesty, integrity, maintenance of high standards of quality, leadership, support of the public interest, and lifelong learning [ACM, IEEE, AITP, SEEPP].
- They sponsor established subgroups such as the Special Interest Group on Computers and Society (SIGCAS) and the Society on Social Implications of Technology (SSIT) that focus directly on ethical and professional issues [SIGCAS, SSIT].
- They develop and refine curricular guidelines such as the ones in this report and its predecessors.
- They participate in the development of accreditation guidelines that ensure the inclusion of professional practice in the curriculum.
- They support the formation of student chapters that encourage students to develop a mature attitude toward professional practice.
- They provide opportunities for lifelong professional development through technical publications, conferences, and tutorials.

Some of these professional codes include the following.

- 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 *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 in making decisions in their careers.

5.5.2 Education and professional practice

Computer engineering programs should inform both students and society about what they can and should expect from people professionally trained in the computing disciplines. Students, for example, need to understand the importance of professional conduct on the job and the ramifications of negligence. They also need to recognize that the professional societies, through their established subgroups emphasizing professional practice and their codes of ethics, can provide a support network that enables them to stand up for what is ethically and professionally right. By laying the groundwork for this support network as part of a four-year program, students can avoid the sense of isolation that young professionals often feel and be well equipped to practice their profession in a mature and ethical way.

Although tenets of professional practice focus on the specific purposes of a society, common themes pervade all of them. Fundamental to all professional practice are the responsibilities of the computing and engineering professional to the public and to the public good. Additionally, professional practice and related ethical codes address conflicts of interest, scope of competence, objectiveness and truthfulness, deception, and professional conduct.

The precepts delineated within professional practice should be the hallmark of all practicing computer engineers. Computer engineers should adopt the tenets of 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, the engineering professions, and humanity.

Chapter 6

Curriculum Implementation Issues

The creation of a complete degree program, that is, an entire program of study, is far from straightforward. The body of knowledge (BoK) 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 body of knowledge plus elective material into required and optional courses, determining required mathematics and science courses, and more general considerations such as creating an overall style or ethos for a specific computer engineering degree program.

6.1 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 the specific character of the program at the beginning 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) compared to more recent material. Currently, 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 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, they may become frustrated and disillusioned with the program.

New engineering areas 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 an individual degree program. These considerations include:

- local needs (institutional or regional)
- needs of an increasingly diverse student population
- interests and background of the faculty

This report promotes flexibility in the design of computer engineering programs by not prescribing how to package core material into courses and leaving room for elective and innovative courses. The core of the body of knowledge provides breadth across the discipline with more in-depth knowledge in key areas. This provides opportunities for students to specialize, while ensuring that graduates are prepared for professional practice in a dynamic and rapidly-evolving field.

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 specializations in areas such as computer communications, embedded computer systems, system level integration, mobile computing systems, computer systems design, computer devices, digital signal processing, multimedia systems, computing and broadcasting, pervasive computing, high integrity computing systems, and real-time systems. It is also possible to develop specializations in such related fields as robotics, biomedical engineering, bioinformatics, digital and analog control systems, and cybersecurity.

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 areas for computer engineering students.
- Additional possibilities also exist depending on local arrangements and circumstances.

6.2 Principles for Curriculum Design

Although 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. 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, a curriculum must teach students to respond to change as well. Computer engineering graduates must keep up to date with modern developments with the expectation of stimulating their engineering curiosity. One of the most important goals of a computer engineering program is to produce students who become lifelong learners.
- *The outcomes expected from a program 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 focus increasing attention on the definition of goals and assessment strategies. Educators who seek to defend the effectiveness of their programs must be able to demonstrate that their curricula in fact accomplish what was intended.
- *Across the board, the curriculum should maintain a consistent ethos that promotes innovation, creativity, and professionalism.* Students respond best when they understand what is expected of them. It is unfair to students to encourage certain 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.

6.3 Basic Computer Engineering Curriculum Components

In assembling the curriculum, institutions must package material into modules, typically into classes or courses. Even though there are many ways for arranging these courses into a complete curriculum, there are some commonalities that can be discussed here as general guidelines.

6.3.1 Coverage of the BoK Core

It is tempting (and easy) to view the BoK itself as a list of courses, and check off coverage of a BoK area by assigning a course or a course sequence to it. In fact, most curriculums will have courses and course sequences that map well to BoK areas. However, coverage of a BoK area can also be done by spreading the knowledge units among several courses. The Computer Science Curricula 2013 in fact recommends that information security be integrated throughout the curriculum. Knowledge units from the Preparation for Professional Practice knowledge areas could similarly be integrated into several courses across a curriculum although it is clear that spreading knowledge unit coverage among several courses takes more coordination and planning for the faculty responsible for those courses.

6.3.2 Course Arrangement

Course sequences are typically used for subject coverage, with the sequence following introductory, intermediate, and advanced areas of study as the student progresses along the curriculum. Examples of introductory computer engineering courses are basic programming courses, digital logic, basic computer organization, and electrical circuits that provide the foundation required for further study. Intermediate courses generally add depth and expose students to state of the art techniques in these areas. Intermediate courses also reinforce knowledge from introductory courses, for example, by requiring a programming language taught in an introductory course to be used to complete work in the intermediate course. Advanced courses are those that require a broad range of knowledge from several courses, such as a computer networks course or an operating systems course. Computer engineering curriculums should also have at least one or two technical elective courses that allow students some choice in shaping their educational experience. To emphasize individual choice, options for technical elective courses should not be overly restrictive, for example, by allowing any upper level course from computer science or electrical and computer engineering to serve as a technical elective.

6.3.3 Laboratory Experiences

Laboratory experiences are crucial to developing skills required to practice the art of computer engineering. Good laboratory experiences force students to demonstrate their understanding of basic knowledge by successfully creating a program or hardware system to meet a set of criteria. Good laboratory experiences also expose students to state-of-the-art tools and methodologies that will prepare them for real engineering work after graduation. The CE2016 steering committee recognizes that lab experience development requires significant faculty time and departmental financial support, but emphasizes that these investments are needed to produce quality computer engineering graduates. Sometimes innovative approaches are needed for adding lab experiences, perhaps by taking a traditional three-hour semester lecture course and transforming it into a two lecture + one-hour lab experience.

6.3.4 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.

6.3.5 Engineering Professional Practice

Professional practice is important for developing well-rounded computer engineering graduates and Chapter 5 offers several methods for incorporating elements of it into a computer engineering curriculum.

6.3.6 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 developing these skills, it is essential that appropriate communication requirements be included within computer engineering courses. Inclusion of communication requirements within computer engineering courses ensures that students can communicate discipline-specific content and contributes to the students' learning of technical material.

6.4 Course Material Presented by Other Departments

In addition to the technical courses specifically on computer engineering, other courses offer material required within the curriculum. For example, computer engineering students must study mathematics and science that form the basis for engineering. In this subsection, we discuss various materials that students must learn, but that typically appear in courses outside of the department in which computer engineering resides. In some cases, students may have learned this material prior to entering the computer engineering program.

6.4.1 Mathematics Requirements

Mathematical techniques and formal mathematical reasoning are integral to most areas of computer engineering and 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 may be taught by computer engineering faculty. 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, one or more courses in discrete structures are important for all students in computer engineering—the steering committee 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 CE2004 report contains the two mathematics knowledge areas of discrete structures and probability and statistics. To this, the CE2016 BoK adds two additional mathematics knowledge areas—analysis of continuous functions and linear algebra. Most computer engineering programs already include coverage of these knowledge areas, and the inclusion now is simply an acknowledgement of that fact. The reasoning for including these four mathematics knowledge areas is offered here.

- *Discrete structures*: All students need knowledge of the mathematical principles of discrete structures and their related tools. All programs should include enough exposure to this area to cover the core learning outcomes specified in the computer engineering body of knowledge.
- *Analysis of continuous functions*: The calculus and differential equations are required to support such computer engineering material as communications theory, signals and systems, and analog electronics—the analysis of continuous functions is fundamental to all engineering programs.
- *Probability and statistics*: This mathematical area underpins considerations of reliability, safety, performance, 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.
- *Linear algebra*: Linear algebra is required for solving networks of equations describing voltage/current relationships in basic circuits, and is used in computer engineering application areas such as computer graphics and robotics.

Students may need to take additional mathematics courses to develop their sophistication in this area and to support additional studies such as communications theory, security, signals and systems, analog electronics, and artificial intelligence. The additional mathematics might consist of courses in any number of areas, including further calculus, transform theory, numerical methods, complex variables, geometry, number theory, optimization methods, or symbolic logic. The choice should depend on program objectives, institutional requirements, and the needs of the individual student.

6.4.2 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 natural 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 natural 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 hands-on experience with hypothesis formulation, experimental design, hypothesis testing, and data analysis. While a curriculum may provide this experience as part of the natural 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 natural sciences underlying computer engineering and relevant application areas.
- Students must develop an understanding of the scientific method and gain experience in this mode of inquiry through 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.

6.4.3 Other Requirements

Many institutions have additional requirements that apply to all students, such as general education requirements. The size and content of these requirements vary widely, depending on the home country, the institutional mission, legal requirements, and other factors. Courses to satisfy these requirements 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.

6.5 Sample Curricula

Appendix B provides sample curricula 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 four to 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.

Institutions in the United States use the above pattern; the same is true for many other parts of the world. The fourth example of a curriculum implementation represents a typical four-year program in China. The fifth example reflects curricula from the Bologna Declaration in Europe.

Chapter 7

Institutional Adaptations

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 ways to address these issues. For schools with existing engineering programs, much of what follows may already be in place or understood.

7.1 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 an individual institution and the interests and expertise of its faculty members. 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.

- *Type of institution and the expectations for its degree programs:* Institutions vary enormously in mission, 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.
- *Spectrum of computer engineering:* Each institution needs to select a focus for its program, ensuring the ethos of computer engineering with proper balance between breadth and depth.
- *Range of postgraduate options that students pursue:* An institution whose primary purpose is to prepare a skilled workforce for the computer engineering profession is likely to have different curricular goals than one seeking to prepare students for research and graduate study. Each individual school must ensure that the curriculum it offers allows students the necessary preparation for their eventual academic and career paths including those outside computer engineering.
- *Preparation and background of entering students:* Students at different institutions—and often within a single institution—vary substantially in their level of preparation. because of this, computer engineering departments often need to tailor their introductory offerings so that they meet the needs of their students.
- *Position of the program within the institution:* Computer engineering programs may reside in schools (colleges) of computing, schools of engineering, and/or schools of arts and sciences, for example. In any case, these programs require a supportive environment that will ensure the ongoing health and vitality of the program.
- *Faculty resources:* The number of faculty members supporting a computer engineering program may vary from fewer than five at a small college to a hundred or more at a large research university. Program size heavily influences the flexibility and options available to a program. Independent of the program size, faculty members need to set priorities for ways in which they will use their limited resources.
- *Interests and expertise of the faculty:* Individual curricula often vary due to the specific interests and knowledge base of the department, particularly at smaller institutions where expertise is concentrated in specific areas.

Creating a workable curriculum requires finding an appropriate balance among these factors, a balance 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.

7.2 Attracting and retaining faculty

One of the most daunting problems that computer engineering departments face is the problem of attracting, and then retaining, 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, make it difficult to attract and retain faculty. Institutions will need to have an aggressive plan to both recruit and retain faculty; incentives such as hiring packages and modified teaching responsibilities may prove advantageous for this endeavor. Active participation in professional organizations provides networking opportunities with leaders of peer programs, which in turn may result in greater visibility and access to potential faculty candidates. Other possible strategies include collaborative and/or interdisciplinary efforts with other programs and/or institutions.

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, sabbaticals (perhaps in industry), consulting, or other activities.

7.3 The need for adequate laboratory resources

It is essential for institutions to recognize that the financial resources required to support a computer engineering program are significant. Software acquisition and maintenance can represent a substantial fraction of the overall cost of computing, particularly if one includes the development costs of courseware. Acquisition and maintenance of the hardware and instrumentation infrastructure required for experimentation and hands-on system development by students is costly. 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 curricula typically include many required laboratories. 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. Close contacts with relevant industries can lead to the ready availability of interesting and up-to-date case study material, and also can offer opportunities for students to engage in internships. Refreshing laboratory material on a regular basis serves to continually motivate and excite new students.

Finally, with the availability of up-to-date reference materials on the internet, institutions should provide access to such resources as the IEEE Xplore Digital Library and the ACM Digital Library. Webinars, e-books, online tutorials, MOOCs, and other resources are all increasingly available and relevant; these are available through, for instance, the ACM Learning Center.

7.4 Transfer and educational pathways

Multiple factors such as admission criteria, student academic preparedness, the diversity of student population, life experiences, and different configurations of educational pathways shape access, retention, and degree

attainment across undergraduate computer engineering programs. The transition points from K-12 through college education vary for successful employment or for further education with advanced degrees throughout the world. This section examines pathways into and through undergraduate CE degree programs from a global perspective.

7.4.1 Four-year transfers

Understanding the entry points into, and the pathways through, undergraduate CE programs will help structure these pathways to serve students, companies, and the computing industry. Specifically, appreciating variations and compatibilities in these pathways across the regions of the world will help enable and support a global computing workforce.

The steering committee considered pathways toward an undergraduate program in computer engineering and considered the rigor of some CE programs. The group defined a "demanding" program as one that requires a significant number of science courses beyond general education, at least five courses in mathematics, and one in which more than 50% of the program relates technically to computer engineering. It hypothesized that fewer students transfer into demanding programs as compared with transfers into non-demanding programs. If a program is too demanding students may be less likely to complete the program successfully. Transfers would likely take place only between universities of equal caliber. There are many exceptions, of course. Notwithstanding, one would expect that successful transfers between two demanding CE programs and transfers between two non-demanding CE programs would be highly more likely than a transfer from a non-demanding CE program to a demanding CE program.

7.4.2 Technical institute transfers

It would be unlikely that a person enrolled in a program at a technical institute could successfully transfer to four-year computer engineering program at a university. The contexts of the two programs would be very different. Students attending technical institutes would likely not have studied mathematics and science at university levels. Additionally, those students would likely not have the requisite general education courses expected from this experience. Hence, course transfer would likely not occur.

In many parts of the world, transfer from technical institutes to university CE programs is almost non-existent. Although exceptions could occur, it is almost impossible for a student to transfer any course experiences at a technical institute to a university.

7.4.3 Community college transfers

In countries where community colleges or two-year college programs thrive, transfer to a university CE program from a community college is common. In fact, such a mode of transfer is even encouraged, especially in the United States and Canada. In the United States, for example, students in some states have a legal right to transfer credit for the same course from a community college to a university program. In fact, the two courses might even have the same code and title such as CHEM 101: Chemistry 1. Indeed, many states distinguish community college transfer programs by sponsoring the Associate in Science (A.S.) degree as opposed to the Associate in Applied Science (A.A.S.) degree, which is a career-oriented degree.

Going a step further, it is very common in the United States to have *articulation agreements* between community colleges and universities. These agreements have a certain legal status in that a student who successfully passes courses at a community college has a right to transfer that course to a university that is a signatory to the articulation agreement. Since student enrollment at community colleges is approximately equal to the undergraduate student enrollment at universities, this vehicle of study is very popular around the country. It is

common for a community college to have articulation agreements with several universities to which student transfer is likely.

Appendix A

Computer Engineering Body of Knowledge

This appendix to the *Computing Curricula - Computer Engineering (CE2016)* 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 Chapter 3 of the report. Included with this appendix is a summary of the fundamental concepts that are necessary to understand the recommendations. For the benefit of the reader, we repeat some of the material that already appears in the body of the report. Consult Chapter 3 for more details.

A.1 Introduction

The CE2016 steering committee developed this model curriculum by first defining the primary disciplines that make up the body of knowledge for computer engineering. The areas that contain material that should be included in all computer engineering curricula are as follows.

CE-CAE	Circuits and Electronics
CE-CAL	Computing Algorithms
CE-CAO	Computer Architecture and Organization
CE-DIG	Digital Design
CE-ESY	Embedded Systems
CE-NWK	Computer Networks
CE-PPP	Preparation for Professional Practice
CE-SEC	Information Security
CE-SGP	Signal Processing
CE-SPE	Systems and Project Engineering
CE-SRM	System Resource Management
CE-SWD	Software Design

A.2 Structure of the Body of Knowledge

The computer engineering body of knowledge has a three-level hierarchical structure. The highest level of the hierarchy is the knowledge area (KA), which represents a specific disciplinary subfield, *not* a course. Knowledge areas contain an "area scope" that describes the context of the specific knowledge area. The knowledge areas are broken down into smaller divisions called knowledge units (KUs), which represent individual themes within an area. We then describe each knowledge unit by a set of learning outcomes, which represent the lowest level of the hierarchy.

A.2.1 Core and supplementary components

For this CE2016 document, the goal is to keep the required component of the body of knowledge as small as possible. We do this to allow programs in computer engineering to be as flexible as possible. To implement this principle, we made a distinction among the KUs by identifying those that are core or essential units in the curriculum compared to those that are supplementary or extra units. Core components comprise knowledge and

skills for which there is broad consensus that anyone obtaining a four-year degree in the field should acquire. Supplementary components comprise knowledge and skills that reflect expectations for advanced work per the needs of a program.

The steering committee thinks it helpful to emphasize the following points.

- The core components refer to the knowledge and skills all students in all computer engineering degree programs *should* attain. Several learning outcomes that are important in the education of many students are not included as core and appear as supplementary. Absence of some learning outcomes among the core components does not imply a negative judgment about their value, importance, or relevance. Rather, it simply means that the learning outcome is not a requirement of *every* student in *all* CE degree programs.
- The core components do *not* constitute a complete curriculum. Additional technical areas, as well as supporting mathematics, science, and general studies, are necessary to produce a competent computer engineer.
- It is not the case that a program should achieve core knowledge units only within a set of introductory courses early in the four-year curriculum. Many core knowledge units are indeed introductory. However, a program can address some core KUs only after students have developed significant background in their studies.

A.2.2 Assessing the time required to cover a unit

To give readers a sense of the time required to cover a specific unit, this report follows the same pattern used in other curricula reports. The CE2016 steering committee has chosen to express time in hours, specifically in core hours. This corresponds to the in-class time required to present material in a traditional lecture-oriented format. Hence, we define one “core hour” or one lecture hour as one 50-minute period.

To dispel any potential confusion, however, it is important to underscore the following observations about the use of lecture hours as a measure.

- This report does not seek to endorse the lecture format. Even though we have used a metric with its roots in a classical, lecture-oriented form, we believe that other styles can be at least as effective, particularly given recent improvements in educational technology. These include forms such as flipped classrooms, massive open online courses (MOOCs), blended learning, pre-recorded lectures, and seminars. 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 metric, in the sense that five core hours will presumably take approximately five times as much time or effort to address as one core hour, independent of the teaching style.
- The hours specified do not include time spent outside of class. The time assigned to a knowledge 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 is approximately three times the in-class time. Thus, a unit listed as requiring three hours typically entails a total of twelve hours (three in class and nine outside class).
- The hours listed for a knowledge unit represent a minimum level of coverage. Users should interpret the time measurements we have assigned for each knowledge unit as a minimal amount of time necessary to enable a student to achieve related learning outcomes for that unit. Many instructors will find that delivery of material to the level of depth that they wish to incorporate will take much longer than this; it is always appropriate to spend more time on a unit than the recommended minimum.

A.2.3 Tags for KAs and KUs

We identify a knowledge area with a tag such as CE-NWK representing the “Computer Networks” knowledge area for computer engineering. We identify each knowledge unit by adding a numeric suffix to the area identification; as an example, CE-NWK-2 is the second knowledge unit for the knowledge area of computer networks.

Supplementary knowledge units have only elective learning outcomes and they do not contain any recommended core hours.

A.2.4 Common KUs

Within each knowledge area, the first KU is "History and overview" and the second is "Relevant tools, standards and/or engineering constraints." These two KUs provide context for the rest of the KA. And the first KU—history and overview—also provides context for the learning outcomes, including important contributors to, and developments in, the area. Engineering practice requires the use of modern tools and contemporary standards, which will change over time. The extent of these KUs vary greatly by knowledge area and the goals of the program.

A.3 Learning Outcomes

To capture the sense of what students should learn in each knowledge unit, this report uses *learning outcomes* to describe each knowledge unit. The emphasis on *learning* is important. Taxonomies of verbs such as "define" or "evaluate" are useful to describe the expected depth of learning. Levels of learning range from basic abilities, such as reciting definitions, to advanced abilities, such as engaging in synthesis and evaluation. Hence, *learning outcomes* provide a mechanism for describing not just knowledge and relevant practical skills, but also personal and transferable skills. They describe what we expect a student can do or know by the time of graduation. We can infer the minimal desired depth of coverage associated with each knowledge unit from the language used to express the learning outcomes. Learning outcomes can be associated with a knowledge unit, a class activity, a course, or even a degree program.

In this report, the steering committee has tried to limit the number of learning outcomes to emphasize essential knowledge and skills. Programs may choose to structure the curriculum so that students demonstrate their attainment of knowledge and skills in a wide variety of ways. Imaginative approaches to assessing learning outcome attainment can lead to a unique expression of a range of skills in well-conceived assignments.

A.4 Summary of the CE body of knowledge

Table A.1 lists the twelve knowledge areas that form the CE body of knowledge areas as presented in this report, together with their associated knowledge units. This is the CE body of knowledge. The table also shows the core hours (core lecture hours) associated with each area and each unit. For example,

CE-ESY-5 Parallel input and output [3]

indicates that "parallel input and output" should have a relative emphasis measured by three core lecture hours and it is the fifth knowledge unit of the "embedded systems" knowledge area, which is core for a computer engineering degree program, with a relative emphasis measured by three core lecture hours. The absence of a number such as [3] means the KU is not core; therefore, it is supplementary.

Table A-1: CE2016 Body of Knowledge
(CE Core Hours: 420)

Computer Engineering Knowledge Areas and Units

CE-CAE	**Circuits and Electronics** [50 core hours]
CE-CAE-1	History and overview [1]
CE-CAE-2	Relevant tools, standards, and/or engineering constraints [3]
CE-CAE-3	Electrical quantities and basic elements [4]
CE-CAE-4	Electrical circuits [11]
CE-CAE-5	Electronic materials, diodes, and bipolar transistors [7]
CE-CAE-6	MOS transistor circuits, timing, and power [12]
CE-CAE-7	Storage cell architecture [3]
CE-CAE-8	Interfacing logic families [3]
CE-CAE-9	Operational amplifiers [3]
CE-CAE-10	Mixed-signal circuit design [3]
CE-CAE-11	Design parameters and issues
CE-CAE-12	Circuit modeling and simulation methods

CE-CAL	**Computing Algorithms** [30 core hours]
CE-CAL-1	History and overview [1]
CE-CAL-2	Relevant tools, standards and/or engineering constraints [1]
CE-CAL-3	Basic algorithmic analysis [4]
CE-CAL-4	Algorithmic strategies [6]
CE-CAL-5	Classic algorithms for common tasks [3]
CE-CAL-6	Analysis and design of application-specific algorithms [6]
CE-CAL-7	Parallel algorithms and multi-threading [6]
CE-CAL-8	Algorithmic complexity [3]
CE-CAL-9	Scheduling algorithms
CE-CAL-10	Basic computability theory

CE-CAO	**Computer Architecture and Organization** [60 core hours]
CE-CAO-1	History and overview [1]
CE-CAO-2	Relevant tools, standards and/or engineering constraints [1]
CE-CAO-3	Instruction set architecture [10]
CE-CAO-4	Measuring performance [3]
CE-CAO-5	Computer arithmetic [3]
CE-CAO-6	Processor organization [10]
CE-CAO-7	Memory system organization and architectures [9]
CE-CAO-8	Input/Output interfacing and communication [7]
CE-CAO-9	Peripheral subsystems [7]
CE-CAO-10	Multi/Many-core architectures [5]
CE-CAO-11	Distributed system architectures [4]

CE-DIG	**Digital Design** [50 core hours]
CE-DIG-1	History and overview [1]
CE-DIG-2	Relevant tools, standards, and/or engineering constraints [2]
CE-DIG-3	Number systems and data encoding [3]
CE-DIG-4	Boolean algebra applications [3]
CE-DIG-5	Basic logic circuits [6]
CE-DIG-6	Modular design of combinational circuits [8]
CE-DIG-7	Modular design of sequential circuits [9]
CE-DIG-8	Control and datapath design [9]
CE-DIG-9	Design with programmable logic [4]
CE-DIG-10	System design constraints [5]
CE-DIG-11	Fault models, testing, and design for testability

CE-ESY	**Embedded Systems** [40 core hours]
CE-ESY-1	History and overview [1]
CE-ESY-2	Relevant tools, standards, and/or engineering constraints [2]
CE-ESY-3	Characteristics of embedded systems [2]
CE-ESY-4	Basic software techniques for embedded applications [3]
CE-ESY-5	Parallel input and output [3]
CE-ESY-6	Asynchronous and synchronous serial communication [6]
CE-ESY-7	Periodic interrupts, waveform generation, time measurement [3]
CE-ESY-8	Data acquisition, control, sensors, actuators [4]
CE-ESY-9	Implementation strategies for complex embedded systems [7]
CE-ESY-10	Techniques for low-power operation [3]
CE-ESY-11	Mobile and networked embedded systems [3]
CE-ESY-12	Advanced input/output issues [3]
CE-ESY-13	Computing platforms for embedded systems

CE-NWK	**Computer Networks** [20 core hours]
CE-NWK-1	History and overview [1]
CE-NWK-2	Relevant tools, standards, and/or engineering constraints [1]
CE-NWK-3	Network architecture [4]
CE-NWK-4	Local and wide area networks [4]
CE-NWK-5	Wireless and mobile networks [2]
CE-NWK-6	Network protocols [3]
CE-NWK-7	Network applications [2]
CE-NWK-8	Network management [3]
CE-NWK-9	Data communications
CE-NWK-10	Performance evaluation
CE-NWK-11	Wireless sensor networks

CE-PPP	**Preparation for Professional Practice** [20 core hours]
CE-PPP-1	History and overview [1]
CE-PPP-2	Relevant tools, standards, and/or engineering constraints [1]
CE-PPP-3	Effective communication strategies [2]
CE-PPP-4	Interdisciplinary team approaches [1]
CE-PPP-5	Philosophical frameworks and cultural issues [2]
CE-PPP-6	Engineering solutions and societal effects [2]
CE-PPP-7	Professional and ethical responsibilities [3]
CE-PPP-8	Intellectual property and legal issues [3]
CE-PPP-9	Contemporary issues [2]
CE-PPP-10	Business and management issues [3]
CE-PPP-11	Tradeoffs in professional practice

CE-SEC	**Information Security** [20 core hours]
CE-SEC-1	History and overview [2]
CE-SEC-2	Relevant tools, standards, and/or engineering constraints [2]
CE-SEC-3	Data security and integrity [1]
CE-SEC-4	Vulnerabilities: technical and human factors [4]
CE-SEC-5	Resource protection models [1]
CE-SEC-6	Secret and public key cryptography [3]
CE-SEC-7	Message authentication codes [1]
CE-SEC-8	Network and web security [3]
CE-SEC-9	Authentication [1]
CE-SEC-10	Trusted computing [1]
CE-SEC-11	Side-channel attacks [1]

Computer Engineering Knowledge Areas and Units	
CE-SGP	**Signal Processing** [30 core hours]
CE-SGP-1	History and overview [1]
CE-SGP-2	Relevant tools, standards, and/or engineering constraints [3]
CE-SGP-3	Convolution [3]
CE-SGP-4	Transform analysis [5]
CE-SGP-5	Frequency response [5]
CE-SGP-6	Sampling and aliasing [3]
CE-SGP-7	Digital spectra and discrete transforms [6]
CE-SGP-8	Finite and infinite impulse response filter design [4]
CE-SGP-9	Window functions
CE-SGP-10	Multimedia processing
CE-SGP-11	Control system theory and applications
CE-SPE	**Systems and Project Engineering** [35 core hours]
CE-SPE-1	History and overview [1]
CE-SPE-2	Relevant tools, standards and/or engineering constraints [3]
CE-SPE-3	Project management principles [3]
CE-SPE-4	User experience* [6]
CE-SPE-5	Risk, dependability, safety and fault tolerance [3]
CE-SPE-6	Hardware and software processes [3]
CE-SPE-7	Requirements analysis and elicitation [2]
CE-SPE-8	System specifications [2]
CE-SPE-9	System architectural design and evaluation [4]
CE-SPE-10	Concurrent hardware and software design [3]
CE-SPE-11	System integration, testing and validation [3]
CE-SPE-12	Maintainability, sustainability, manufacturability [2]
CE-SRM	**Systems Resource Management** [20 core hours]
CE-SRM-1	History and overview [1]
CE-SRM-2	Relevant tools, standards, and/or engineering constraints [1]
CE-SRM-3	Managing system resources [8]
CE-SRM-4	Real-time operating system design [4]
CE-SRM-5	Operating systems for mobile devices [3]
CE-SRM-6	Support for concurrent processing [3]
CE-SRM-7	System performance evaluation
CE-SRM-8	Support for virtualization
CE-SWD	**Software Design** [45 core hours]
CE-SWD-1	History and overview [1]
CE-SWD-2	Relevant tools, standards, and/or engineering constraints [3]
CE-SWD-3	Programming constructs and paradigms [12]
CE-SWD-4	Problem-solving strategies [5]
CE-SWD-5	Data structures [5]
CE-SWD-6	Recursion [3]
CE-SWD-7	Object-oriented design [4]
CE-SWD-8	Software testing and quality [5]
CE-SWD-9	Data modeling [2]
CE-SWD-10	Database systems [3]
CE-SWD-11	Event-driven and concurrent programming [2]
CE-SWD-12	Using application programming interfaces
CE-SWD-13	Data mining
CE-SWD-14	Data visualization

* User experience (UX) was formerly known as human-computer interaction (HCI)

A.5 Knowledge Areas and Knowledge Units

CE-CAE Circuits and Electronics

[50 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. Purpose and role of circuits and electronics in computer engineering, including key differences between analog and digital circuits, their implementations, and methods of approximating digital behavior with analog systems
2. Definitions and representations of basic electrical quantities and elements, as well as the relationships among them
3. Analysis and design of simple electronic circuits using appropriate techniques, including software tools, and incorporating appropriate constraints and tradeoffs
4. Properties of materials that make them useful for constructing electronic devices
5. Properties of semiconductor devices, their use as amplifiers and switches, and their use in the construction of a range of basic analog and logic circuits
6. Effects of device parameters and various design styles on circuit characteristics, such as timing, power, and performance
7. Practical considerations and tradeoffs associated with distributing signals within large circuits and of interfacing between different logic families or with external environments

CE-CAE Core Knowledge Units

CE-CAE-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Describe ways in which computer engineering uses or benefits from electronic devices and circuits.
- Identify some contributors to circuits and electronics and relate their achievements to this knowledge area.
- Explain the key differences between analog and digital systems, their implementations, and methods for approximating digital behavior with analog systems.
- Summarize basic electrical quantities and elements that show the relationship between current and voltage.
- Describe the use of the transistor as an amplifier and as a switch.
- Explain the historical progression from discrete devices to integrated circuits to current state-of-the-art electronics.

CE-CAE-2 Relevant tools, standards, and/or engineering constraints

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Describe at least two common types of circuit simulators and contrast the advantages and applications of each.
- Interpret issues associated with interfacing digital computer systems with an analog world, including the use of standard data conversion circuits.
- Summarize the role of standards in compatibility, interconnection, and safety of systems.
- Articulate the purpose of buses and other interconnection and communication networks.
- Illustrate the role of constraints, parameters, and tradeoffs in electronic circuit design.

CE-CAE-3 Electrical quantities and basic elements

Minimum core coverage time: 4 hours

Core Learning Outcomes:

- State the definitions and representations of basic electrical quantities (charge, current, voltage, energy, power), as well as the relationships among them.
- Define and represent basic circuit elements (resistors, inductors, capacitors).
- Solve problems using Ohm's law, including its power representations.
- Analyze basic electrical circuits using Ohm's law.
- Explain the difference between resistance and reactance, the meaning of phase, and the effect of frequency on capacitance and inductance.

Elective Learning Outcomes:

- Interpret the role of capacitors and inductors as basic storage elements.
- Contrast related electrical quantities and concepts including frequency response, sinusoids, convolution, diodes and transistors, and other storage elements.
- Provide examples of using circuit simulators to model and analyze simple circuits.

CE-CAE-4 Electrical circuits

Minimum core coverage time: 11 hours

Core Learning Outcomes:

- Contrast various elements of circuit models including independent and dependent sources as well as series and parallel elements.
- Analyze basic electrical circuits using mesh and nodal analysis, Kirchoff's laws, superposition, Thevenin's theorem, and Norton's theorem.
- Apply properties of circuits containing various combinations of resistance (R), inductance (L), and capacitance (C) elements including time constants, transient and steady-state responses, and damping.
- Analyze and design simple circuits containing R, L, and C elements.
- Illustrate the frequency domain characteristics of electrical circuits.
- Contrast power for resistive and reactive circuits.
- Define and use the phasor representations of voltage and current in analyzing circuits.
- Calculate the response of electrical circuits from sinusoidal signal excitation.
- Define and use impedance and admittance as well as source transformations.
- Model and analyze simple resistive and RLC circuits using a circuit simulator.

Elective Learning Outcomes:

- Identify the characteristics and uses of transformers.
- Explain the relation between electrical quantities and concepts such as transfer functions, two-port circuits, parallel and series resonance, maximum power transfer, and mutual inductance.
- Describe the characteristics of electronic voltage sources such as ideal voltage source, voltage references, emitter followers, and voltage sources utilizing operational amplifiers.
- Express the characteristics of electronic current sources for the following: ideal current source; transistor current sources; common-emitter, cascode, and regulated cascode circuits; current sources utilizing operational amplifiers.

CE-CAE-5 Electronic materials, diodes, and bipolar transistors

Minimum core coverage time: 7 hours

Core Learning Outcomes:

- Explain characteristics and properties of electronic materials including electrons and holes; doping, acceptors, and donors; p-type and n-type materials; conductivity and resistivity; drift and diffusion currents, mobility, and diffusivity.
- Illustrate the operation and properties of diodes, including I-V characteristics, regions of operation, equivalent circuit models and their limitations.
- Illustrate the operation and properties of NPN and PNP transistors, including I-V characteristics, regions of operation, equivalent circuit models and their limitations, and transfer characteristic with a load resistor.
- Contrast NPN and PNP transistor biasing for logic and amplifier applications.
- Explain the properties of bipolar transistors when used as amplifiers and as switches.
- Produce mathematical models to represent material properties of electronic devices.
- Provide examples of using mathematical models in circuit simulators.

Elective Learning Outcomes:

- Contrast the Schottky, Zener, and variable capacitance diodes.
- Design a single diode circuit and describe the significance of a load line.
- Illustrate multidiode circuits such as rectifiers and direct current (DC) involving DC-DC voltage level converters.
- Design a multidiode circuit including rectifiers.
- Design a multidiode circuit including DC-DC voltage level converters.
- Implement diode logic using only AND and OR functions.
- Provide examples of bipolar transistors used in the construction of a range of common circuits.

CE-CAE-6 MOS transistor circuits, timing, and power

Minimum core coverage time: 12 hours

Core Learning Outcomes:

- Illustrate the operation and properties of nMOS (n-type metal-oxide semiconductor) and pMOS field-effect transistors, including I-V characteristics, regions of operation, equivalent circuit models and their limitations, enhancement-mode and depletion-mode devices, and transfer characteristic with a load resistor.
- Apply nMOS and pMOS transistor biasing for logic and amplifier applications.

- Contrast the properties of nMOS and pMOS transistors used as switches.
- Implement basic logic functions using nMOS, pMOS, and complementary metal-oxide semiconductor (CMOS) logic.
- Implement logic functions using pass transistors and transmission gates.
- Analyze the implications of implementing logic functions with switch networks versus logic gates.
- Define propagation delay, rise time, and fall time.
- Illustrate simplified Unit-Delay and Tau models for circuit timing.
- Analyze the effects of logic gate fan-in and fan-out on circuit timing and power and their associated tradeoffs.
- Contrast the effects of transistor sizing on timing and power, including nMOS and CMOS power/delay scaling.
- Compute the effects on circuit characteristics of various design styles, e.g., static logic, dynamic logic, multiple clocking schemes.

CE-CAE-7 Storage cell architecture
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Contrast the circuit properties of implementations of various storage elements (e.g., latches, flip-flops, clocked registers).
- Contrast the circuit properties of implementations of various memory cells (e.g., static RAM, dynamic RAM, ROM) and related circuitry (e.g., sense amplifiers).
- Contrast storage elements and memory cells, emphasizing the tradeoffs that make each appropriate for specific uses.

Elective Learning Outcomes:
- Contrast the circuit properties of different kinds of non-volatile storage elements (e.g., flash memory, ROM).
- Derive timing diagrams showing the relationships among input, output, and clock signals for different storage devices.

CE-CAE-8 Interfacing logic families
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Explain the practical difficulties resulting from interfacing signals within a system and to the external world.
- Employ terminal characteristics of various logic families and of standard interfaces.
- Write the requirements for common signal translations between different logic families, such as between transistor-transistor logic (TTL) and CMOS.
- Illustrate common methods to overcome difficulties when interfacing different logic families.
- Explain the practical difficulties resulting from single-ended to differential and differential to single-ended conversions.
- Contrast transmission line characteristics, reflections, and options for bus termination including passive, active, DC, and alternating current (AC) features.

CE-CAE-9 Operational amplifiers
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Interpret the properties of an ideal operational amplifier (op-amp).
- Analyze and design circuits containing ideal op-amp circuits to include inverting and non-inverting amplifiers, summing and difference amplifiers, integrators, and low-pass filters.

Elective Learning Outcomes:
- Contrast the properties of non-ideal op-amps to include DC errors, common-mode rejection ratio (CMRR), input and output resistances, frequency response, output voltage, and current limitations.
- Analyze and design simple circuits containing non-ideal op-amps.
- Contrast and design multistage op-amp circuits.

CE-CAE-10 Mixed-signal circuit design
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Discuss common types of mixed-signal circuits and applications, including digital-to-analog (D/A) and analog-to-digital (A/D) converters and sample-and-hold circuits.
- Describe key characteristics of D/A and A/D converters, such as least-significant bit (LSB), linearity, offset, and gain errors.
- Contrast the properties that distinguish between specific D/A and A/D converters for meeting system design requirements.
- Analyze issues associated with the integration of digital and analog circuits in a single IC or package, including both benefits and challenges.
- Provide examples of commercial mixed-signal devices.

Elective Learning Outcomes:

- Describe how D/A converter characteristics depend upon implementation; examples include weighted resistor, R/2R resistor ladders, weighted current source converters, and delta-sigma converters.
- Describe how A/D converter characteristics depend upon implementation; examples include successive approximation converters, single and dual slope converters, flash converters, and delta-sigma converters.
- Design A/D and D/A converters to meet given criteria using specified implementations.

CE-CAE Supplementary Knowledge Units

CE-CAE-11 Design parameters and issues
Supplementary

Elective Learning Outcomes:

- Calculate the effects of design parameters on switching energy, power-delay product, power dissipation, and noise margin.
- Indicate issues associated with power supply distribution.
- Describe sources of signal coupling and degradation, and their effects on circuit behavior.
- Contrast transmission line effects, particularly for passive, active, DC, and AC terminations.
- Use appropriate design strategies and software tools for power distributions and transmission lines, incorporating element tolerances and tradeoffs.
- Use appropriate design strategies and software tools to minimize noise and other signal degradations in designs.
- Develop methods for worst-case analysis of circuits.
- Explain Monte Carlo analysis and describe tools for using Monte Carlo analysis in circuit design.
- Examine the use of six-sigma design methods for electronic circuits.

CE-CAE-12 Circuit modeling and simulation methods
Supplementary

Elective Learning Outcomes:

- Predict the benefits and drawbacks associated with simulation as a method of circuit analysis.
- Apply simulation methods for DC analysis, AC analysis, transient analysis, and steady-state analysis.
- Identify aspects of circuits that are not readily amenable to simulation.
- Contrast methods and parameters for controlling simulation to include built-in device models, device parameter controls, and device and circuit libraries.

CE-CAL Computing Algorithms

[30 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. Fundamental algorithmic design principles
2. Analysis of algorithmic behavior, including tradeoffs between algorithms
3. Classic algorithms for such common tasks as searching and sorting
4. Design and analysis of application-specific algorithms
5. Characteristics of parallel algorithms

CE-CAL Core Knowledge Units

CE-CAL-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Explain the role of algorithms in a hardware/software system.
- Give examples of applications in which choice of algorithm is a significant design decision.
- Discuss the contributions of pioneers in the field.
- Explain why theory is important.

CE-CAL-2 Relevant tools, standards and/or engineering constraints

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Use library classes and the algorithms available in application code.
- Explain how to find libraries to support applications of interest.

CE-CAL-3 Basic algorithmic analysis

Minimum core coverage time: 4 hours

Core Learning Outcomes:

- Use big O, omega, and theta notation to characterize asymptotic upper, lower, and tighter bounds on time and space complexity of algorithms.
- Determine the time complexity and the space complexity of simple algorithms.
- Measure the performance of an algorithm empirically.
- Explain why time/space tradeoffs are important in computing systems.

CE-CAL-4 Algorithmic strategies

Minimum core coverage time: 6 hours

Core Learning Outcomes:

- Design and implement brute force algorithms.
- Design and implement greedy algorithms.
- Design and implement an algorithm using a divide and conquer strategy.
- Explain how recursive algorithms work.
- Explain why heuristics are useful and give examples of their use.

CE-CAL-5 Classic algorithms for common tasks

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Describe algorithms historically used for searching and sorting.
- Solve problems using efficient sorting algorithms.
- Explain tradeoffs in choice of appropriate algorithm for common tasks.
- Use abstract data types (such as hash tables and binary search trees) in applications involving search.

CE-CAL-6 Analysis and design of application-specific algorithms
Minimum core coverage time: 6 hours

Core Learning Outcomes:
- Identify characteristics of an application that influence algorithm choice.
- Explain features of algorithms used in application domains such as control applications, mobile or location-aware applications, discrete event simulation applications or encryption/decryption algorithms.
- Identify factors having impact on the performance of application-specific algorithms.

CE-CAL-7 Parallel algorithms and multi-threading
Minimum core coverage time: 6 hours

Core Learning Outcomes:
- Analyze the parallelism inherent in a simple sequential algorithm.
- Explain why communication and coordination are critical to ensure correctness.
- Calculate the speedup attainable in theory and explain factors limiting attainable speedup.
- Explain limitations to scalability.
- Discuss parallel algorithm structure and give examples.
- Illustrate ways to manage algorithmic execution in multiple threads.
- Select appropriate methods for measuring the performance of multithreaded algorithms.

CE-CAL-8 Algorithmic complexity
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Explain what it means for a problem to be NP-complete.
- Give examples of NP-complete problems and explain why this is important.
- Apply appropriate heuristics in the context of addressing intractable problems.

CE-CAL Supplementary Knowledge Units

CE-CAL-9 Scheduling algorithms
Supplementary

Elective Learning Outcomes:
- Explain the underlying strategies in scheduling based on priority of the job, the length of the job, arrival time, and the impact of real time constraints.
- Explain factors influencing the choice of a scheduling algorithm in an application.
- Analyze the impact of the scheduling algorithm on system performance.
- Illustrate the performance of a scheduling algorithm given a job set.

CE-CAL-10 Basic computability theory
Supplementary

Elective Learning Outcomes:
- Illustrate and analyze system behavior using finite state machines.
- Explain how regular expressions are related to finite state machines and why this is important.
- Design a deterministic finite state machine to accept a simple language.
- Generate a regular expression to represent a specified language.
- Explain what a context free grammar is and why finite state machines do not recognize all context free languages.
- Explain what an undecidable problem is.
- Discuss what the halting problem is and why it is significant.

CE-CAO Computer Architecture and Organization

[60 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. History of computer architecture, organization, and its role in computer engineering
2. Standards and design tools used in computer architecture and organization
3. Instruction set architectures, including machine and assembly level representations and assembly language programming
4. Computer performance measurement, including performance metrics and benchmarks and their strengths and weaknesses
5. Arithmetic algorithms for manipulating numbers in various number systems
6. Computer processor organization and tradeoffs, including data path, control unit, and performance enhancements
7. Memory technologies and memory systems design, including main memory, cache memory, and virtual memory
8. Input/output system technologies, system interfaces, programming methods, and performance issues
9. Multi/many-core architectures, including interconnection and control strategies, programming techniques, and performance
10. Distributed system architectures, levels of parallelism, and distributed algorithms for various architectures

CE-CAO Core Knowledge Units

CE-CAO-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Identify some contributors to computer architecture and organization and relate their achievements to the knowledge area.
- Articulate differences between computer organization and computer architecture.
- Sketch a block diagram showing the main components of a simple computer.
- Explain the reasons and strategies for different computer architectures and indicate some strengths and weaknesses inherent in each.
- Identify some modern techniques for high-performance computing, such as multi/many-core and distributed architectures.

CE-CAO-2 Relevant tools, standards and/or engineering constraints

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Identify tools to simulate computer systems at different levels of design abstraction: system, instruction set processor (ISP), register-transfer language (RTL), and gate level.
- Discuss the type of information contained in one or more component interconnect standards.
- Discuss how architecture design choices and tradeoffs influence important consequences such as performance and power.

Elective Learning Outcome:

- Contrast two hardware description languages, such as VHDL and Verilog.

CE-CAO-3 Instruction set architecture

Minimum core coverage time: 10 hours

Core Learning Outcomes:

- Explain the organization of a von Neumann machine and its major functional units.
- Illustrate how a computer fetches from memory, decodes, and executes an instruction.
- Articulate the strengths and weaknesses of the von Neumann architecture, compared to a Harvard or other architecture.
- Describe the primary types of computer instructions, operands, and addressing modes.
- Explain the relationship between the encoding of machine-level operations at the binary level and their representation in a symbolic assembly language.
- Explain different instruction format options, such as the number of addresses per instruction and variable-length versus fixed-length formats.
- Describe reduced (RISC) vs complex (CISC) instruction set computer architectures.
- Write small assembly language programs to demonstrate an understanding of machine-level operations.
- Implement some fundamental high-level programming constructs at the assembly-language level, including control flow structures such as subroutines and procedure calls.
- Write small assembly language programs to access simple input/output devices using program-controlled and interrupt-driven methods.

Elective Learning Outcome:

- Describe features and applications of short-vector instruction sets: Streaming extensions, AltiVec, relationship between computer architecture and multimedia applications.

CE-CAO-4 Measuring performance

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- List the factors that contribute to computer performance.
- Articulate the rationale for and limitations of commonly used computer performance metrics, such as clock rate, MIPS, cycles per instruction, throughput, and bandwidth.
- Describe the rationale for and limitations of benchmark programs.
- Name and describe two commonly used benchmarks for measuring computer performance, and contrast two different computer systems using published benchmark results.
- Select the most appropriate performance metrics and/or benchmarks for evaluating a given computer system, for a target application.
- Explain the role of Amdahl's law in computer performance and the ways control and data path design can affect performance.

CE-CAO-5 Computer arithmetic

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Determine the characteristics of commonly used number systems such as range, precision, accuracy, and conditions that lead to arithmetic overflow and underflow, and tradeoffs between characteristics of different number systems.
- Describe the limitations of computer arithmetic and the effects of errors on calculations.
- Describe basic arithmetic algorithms for addition, subtraction, multiplication, and division of integer binary numbers.
- Convert numbers to and from the formats specified by the IEEE 754 standard for floating-point arithmetic.
- Describe algorithms for addition, subtraction, multiplication, and division of floating-point numbers.
- Describe how multi-precision arithmetic is performed in a computer system.
- Discuss the effect of a processor's arithmetic unit on its overall performance.

Elective Learning Outcomes:

- Describe algorithms for higher-complexity functions, such as square roots and transcendental functions.
- Describe saturating arithmetic operations and discuss some applications in which saturating arithmetic would be useful.

CE-CAO-6 Processor organization

Minimum core coverage time: 10 hours

Core Learning Outcomes:

- Discuss the relationship between instruction set architecture and processor organization.
- Contrast tradeoffs between alternative implementations of datapaths for a Von Neumann machine.
- Design a datapath and a hard-wired control unit for a simple instruction set architecture.
- Design arithmetic units for multiplication, division, and floating-point arithmetic.
- Explain basic instruction-level parallelism (ILP) using pipelining, the effect of pipelining on performance, and the major hazards that may occur, including performance penalties resulting from hazards.
- Explain the steps needed to mitigate the effect of pipeline hazards caused by branches.
- Describe common exception and interrupt handling mechanisms used in computer systems.
- Describe the characteristics of superscalar architectures, including multi-issue operation, and in-order and out-of-order execution.
- Describe how each of the functional parts of a computer system affects its overall performance.

Elective Learning Outcomes:

- Discuss the way in which instruction sets have evolved to improve performance—for example, predicated/speculative execution and SIMD support.
- Discuss frequency and power scaling issues and their tradeoffs for processor design.
- Discuss how accelerators (e.g., GPUs, DSPs, FPGAs) can be used to improve performance.
- Discuss how to apply parallel processing approaches to design scalar and superscalar processors.
- Discuss how to apply vector processing techniques to enhance instruction sets for multimedia and signal processing.

CE-CAO-7 Memory system organization and architecture

Minimum core coverage time: 9 hours

Core Learning Outcomes:

- Identify the main types of memory technologies presently in use.
- Design a main memory with specified parameters using given memory devices.

- Discuss how memory performance metrics, such as latency, cycle time, bandwidth, and interleaving, are used to measure the effects of memory on overall system performance.
- Explain the use of memory hierarchy to reduce the effective memory latency in a system.
- Describe common cache memory organizations, explain the use of cache memory to improve performance, and discuss cost-performance tradeoffs of different cache organizations.
- Illustrate mechanisms used to provide cache coherence, invalidation/snooping, and shared/exclusive access control.
- Describe the principles of memory management and virtual memory systems.
- Describe characteristics of current secondary storage technologies, such as magnetic, optical, and solid-state drives.

Elective Learning Outcome:

- Understand how errors in memory systems arise, and illustrate several mechanisms used to resolve them, such as error detecting and error correcting systems, and RAID structures.

CE-CAO-8 Input/output interfacing and communication

Minimum core coverage time: 7 hours

Core Learning Outcomes:

- Draw a block diagram showing how a processor interacts with input/output (I/O) devices, including peripheral addressing (isolated vs memory-mapped) handshaking, and buffering.
- Explain the use of interrupts to implement I/O control and data transfers, including vectored and prioritized interrupts, and discuss factors that contribute to interrupt overhead and latency.
- Write small interrupt service routines and I/O drivers using assembly language.
- Illustrate the use of direct memory access (DMA) to interact with IO devices.
- Determine tradeoffs between program-controlled IO, interrupt-driven IO, and DMA for a given application.
- Describe the characteristics of a parallel bus, including data transfer protocols.
- Describe characteristics of asynchronous and synchronous serial communication protocols.
- Discuss tradeoffs between parallel and serial data transmission between devices.

CE-CAO-9 Peripheral subsystems

Minimum core coverage time: 7 hours

Core Learning Outcomes:

- Contrast the characteristics of one or more computer system expansion buses.
- Select an appropriate bus for connecting given components/subsystems to a computer system.
- Describe data access from a secondary storage device such as a magnetic or solid-state disk drive.
- Explain how storage subsystem interface / controllers function.
- Explain how display subsystems and controllers function.
- Describe other input and output device subsystems (e.g., keyboard, mouse, audio).
- Describe communication subsystems: network controllers, and serial and parallel communication functions.

CE-CAO-10 Multi/Many-core architectures

Minimum core coverage time: 5 hours

Core Learning Outcomes:

- Discuss the performance limitations of single-core processors due to clock-frequency and power walls.
- Describe the basic organization of a multi/many-core, shared memory processor.
- Discuss the benefits of homogeneous vs heterogeneous multi/many-core architectures, and tradeoffs between different architectures.
- Discuss on-chip interconnect networks and memory controller issues.
- Describe how programs are partitioned for execution on multi/many-core processors.
- Articulate current programming techniques, models, frameworks, and languages for multi/many-core processors.

CE-CAO-11 Distributed system architectures

Minimum core coverage time: 4 hours

Core Learning Outcomes:

- Explain the differences and tradeoffs between various distributed system paradigms.
- Explain the impact of granularity and levels of parallelism in distributed systems, including threads, thread-level parallelism and multithreading.
- Describe the topology, degrees of coupling, and other characteristics of several current multiprocessor/multicomputer architectures.
- Describe how the client-server model works in a decentralized fashion.
- Explain how agents work and how they solve simple tasks.
- Articulate current programming techniques, models, frameworks, and languages for distributed, parallel processing.

Elective Learning Outcomes:

- Describe modern implementations of the client-server model, such as cloud-based computing.
- Describe the concept of logical clocks versus physical clocks and show how they affect implementation of distributed systems.
- Contrast simple election and mutual exclusion algorithms and their applicability.
- Describe approaches to design for parallelism, synchronization, thread safety, concurrent data structures.
- Discuss distributed transaction models, classification, and concurrency control.

CE-DIG Digital Design

[50 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. Digital design basics: number representation, arithmetic operations, Boolean algebra, and their realization as basic logic circuits
2. Building blocks: combinational, sequential, memories, and elements for arithmetic operations
3. Hardware Description Languages (HDLs), digital circuit modeling, design tools, and tool flow
4. Programmable logic platforms (e.g., FPGAs) for implementing digital systems
5. Datapaths and control units composed of combinational and sequential building blocks
6. Analysis and design of digital systems including design space exploration, and tradeoffs based on constraints such as performance, power, and cost

CE-DIG Core Knowledge Units

CE-DIG-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Identify some early contributors to digital design and relate their achievements to the knowledge area.
- Discuss applications in computer engineering that benefit from the area of digital design.
- Describe how Boolean logic relates to digital design.
- Enumerate key components of digital design such as combinational gates, memory elements, and arithmetic blocks.

CE-DIG-2 Relevant tools, standards, and/or engineering constraints

Minimum core coverage time: 2 hours

Core Learning Outcomes:

- Describe design tools and tool flow (e.g., design entry, compilation, simulation, and analysis) that are useful for the creation and simulation of digital circuits and systems.
- Discuss the need for standards and enumerate standards important to the area of digital design such as floating-point numbers (IEEE 754) and character encoding (ASCII, Unicode)
- Use one of the standard HDLs (e.g., IEEE 1364/Verilog, IEEE 1076/VHDL) for modeling simple digital circuits.
- Define important engineering constraints such as timing, performance, power, size, weight, cost, and their tradeoffs in the context of digital systems design.

CE-DIG-3 Number systems and data encoding

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Convert signed/unsigned, integer/fixed-point decimal numbers to/from binary/hex representations.
- Perform integer/fixed-point addition/subtraction using binary/hex number representations.
- Define precision and overflow for integer/fixed-point, signed/unsigned, addition/subtraction operations.
- Encode/decode character strings using ASCII and Unicode standards.

CE-DIG-4 Boolean algebra applications

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Define basic (AND, OR, NOT) and derived (e.g., NAND, NOR, XOR) Boolean operations.
- Enumerate Boolean algebra laws and theorems.
- Use basic and derived Boolean operations to evaluate Boolean expressions.
- Write and simplify Boolean expressions by applying appropriate laws and theorems and other techniques (e.g., Karnaugh maps).

CE-DIG-5 Basic logic circuits

Minimum core coverage time: 6 hours

Core Learning Outcomes:

- Describe electrical representations of TRUE/FALSE.
- Describe physical logic gate implementations of basic (AND, OR, NOT) and derived (e.g., NAND, NOR, XOR) Boolean operations.

- Describe the high-impedance condition and logic gate implementation such as a tri-state buffer.
- Implement Boolean expressions using the two-level gate forms of AND-OR, OR-AND, NAND-NAND, NOR-NOR and positive/negative/mixed-logic conventions.
- Implement Boolean expressions using multiple gating levels and positive/negative/mixed-logic conventions.
- Discuss the physical properties of logic gates such as fan-in, fan-out, propagation delay, power consumption, logic voltage levels, and noise margin and their impact on the constraints and tradeoffs of a design.
- Explain the need for a hardware description language (HDL) in digital system design.
- Describe the logic synthesis process that transforms an HDL description into a physical implementation.
- Implement combinational networks using an HDL and generate/verify using appropriate design tools.

CE-DIG-6 Modular design of combinational circuits
Minimum core coverage time: 8 hours

Core Learning Outcomes:
- Describe and design single-bit/multi-bit structure/operation of combinational building blocks such as multiplexers, demultiplexers, decoders, and encoders.
- Describe and design the structure/operation of arithmetic building blocks such adders (ripple-carry), subtractor, shifters, and comparators.
- Describe and design structures for improving adder performance such as carry lookahead and carry select.
- Analyze and design combinational circuits (e.g., arithmetic logic unit, ALU) in a hierarchical, modular manner, using standard and custom combinational building blocks.
- Implement combinational building blocks and modular circuits using an HDL and generate/verify using appropriate design tools.

CE-DIG-7 Modular design of sequential circuits
Minimum core coverage time: 9 hours

Core Learning Outcomes:
- Define a clock signal using period, frequency, and duty-cycle parameters.
- Explain the structure/operation of basic latches (D, SR) and flip-flops (D, JK, T).
- Describe propagation delay, setup time, and hold time for basic latches and flip-flops.
- Describe and design the structure/operation of sequential building blocks such as registers, counters, and shift registers.
- Analyze and create timing diagrams for sequential block operation.
- Enumerate design tradeoffs in using different types of basic storage elements for sequential building block implementation.
- Implement sequential building blocks using an HDL and generate/verify using appropriate design tools.
- Describe the characteristics of static memory types such static SRAM, ROM, and EEPROM.
- Describe the characteristics of dynamic memories.

Elective Learning Outcomes:
- Describe techniques (e.g., handshaking) of asynchronous design, and discuss their advantages (e.g., performance/power in some cases) and design issues (e.g., hazards such as race conditions, lack of tool support).
- Describe the characteristics of advanced memory technologies such as multi-port memories, double data rate (DDR) memories, and hybrid memories (e.g., hybrid memory cube, HMC).

CE-DIG-8 Control and datapath design
Minimum core coverage time: 9 hours

Core Learning Outcomes:
- Describe a digital system that is partitioned into control+datapath and explain the need for control to sequence operations on a datapath.
- Contrast the different types of Finite State Machines (FSMs): e.g., Mealy State Machine, Moore State Machine, and Algorithmic State Machine (ASM).
- Represent FSM operation graphically using a state diagram (e.g., Mealy state diagram, Moore state diagram, or ASM chart).
- Analyze state diagrams and create timing diagrams for FSM operation.
- Compute timing parameters such as maximum operating frequency, setup/hold time of synchronous inputs, clock-to-out propagation delays, pin-to-pin propagate delay for a control+datapath design.
- Design an RTL model of a control+datapath using a HDL and synthesize/verify using appropriate design tools.

Elective Learning Outcomes:
- Discuss clock generation, clock distribution, clock skew in relationship to a control+datapath design.
- Use pipelining to improve the performance of a control+datapath design.
- Discuss applications that require serialization/de-serialization of bit streams, and implement a design that performs serialization/de-serialization.

CE-DIG-9 Design with programmable logic
Minimum core coverage time: 4 hours

Core Learning Outcomes:
- Describe basic elements of programmable logic such as lookup tables (LUTs), AND/OR plane programmable logic, programmable mux logic, and programmable routing.
- Discuss programmable logic architectures such as Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs).
- Describe common features of programmable logic architectures such as hard macros (e.g., adders, multipliers, SRAMs), clock generation support (e.g., PLLs, multiple clock networks), and support for different logic standards.
- Implement a digital system in an FPGA or CPLD and describe and evaluate tradeoffs for implementation characteristics such as programmable logic resources that are used, maximum clock frequency, setup/hold times for external inputs, and clock-to-out delay.

Elective Learning Outcomes:
- Describe advanced features of programmable logic architectures in the form of hard macros such as CPUs, high-speed serial transceivers, and support for other transceiver standards (e.g., PCI Express, Ethernet PCS).

CE-DIG-10 System design and constraints
Minimum core coverage time: 5 hours

Core Learning Outcomes:
- Contrast top-down versus bottom-up design methodologies for system design.
- Describe how to use logic synthesis timing constraints with an appropriate design tool for affecting logic generated for a control+datapath implementation.
- Use constraints of clock-cycle latency and clock-cycle throughput to create alternate designs for a digital system.
- Use other appropriate design tools (e.g., power estimator) for design space exploration and tradeoffs based on constraints such as performance, power, and cost.
- Describe the role of testability as a system design constraint and different approaches and tools for improving testability.
- Describe features/architecture of the JTAG standard and its role in digital systems testing.
- Create an HDL-based self-checking behavioral test bench for a digital system design.

CE-DIG Supplementary Knowledge Units

CE-DIG-11 Fault models, testing, and design for testability
Supplementary

Elective Learning Outcomes:
- Explain the need for systematic testing methods in digital design.
- Define fault models such as stuck-at, bridging, and delay.
- Define the terms controllability, observability, test coverage, and test generation when designing a method for testing a digital system.
- Describe design for testability methods such as ad-hoc, full-scan/partial scan and built-in-self-test (BIST).
- Describe the role of computer-aided testing tools for digital systems testing.

CE-ESY Embedded Systems

[40 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. Purpose and role of embedded systems in computer engineering, along with important tradeoffs in such areas as power, performance, and cost
2. Embedded systems software design, either in assembly language or a high-level language or both, for typical embedded systems applications using modern tools and approaches for development and debugging
3. Digital interfacing using both parallel and asynchronous/synchronous serial techniques incorporating typical on-chip modules as such as general purpose I/O, timers, and serial communication modules (e.g., UART, SPI, I2C, and CAN)
4. Analog interfacing using analog-to-digital convertors connected to common sensor elements and digital-to-analog converters connected to typical actuator elements
5. Mobile and wireless embedded systems using both short-range (e.g., Bluetooth, 802.15.4) and long-range (e.g., cellular, Ethernet) in various interconnection architectures

CE-ESY Core Knowledge Units

CE-ESY-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Identify some contributors to embedded systems and relate their achievements to the knowledge area.
- Describe the characteristics of an embedded system and its role in several example applications.
- Explain the reasons for the importance of embedded systems.
- Describe the relationship between programming languages and embedded systems.
- Describe how computer engineering uses or benefits from embedded systems.

CE-ESY-2 Relevant tools, standards, and/or engineering constraints

Minimum core coverage time: 2 hours

Core Learning Outcomes:

- Use an integrated development environment (IDE) to write, compile and/or assemble, and debug a program (high-level or assembly language) for a target embedded system.
- Contrast instrumentation choices for diagnosing/understanding hardware aspects of embedded systems behavior.
- List several standards applicable to embedded such as signaling levels and serial communication protocols.

CE-ESY-3 Characteristics of embedded systems

Minimum core coverage time: 2 hours

Core Learning Outcomes:

- Contrast CPUs used for embedded systems versus those used for general purpose computing.
- Evaluate and rank tradeoffs such as cost, power, and performance for different embedded systems applications.
- Describe architectural features of the target embedded system(s) (register structure, memory architecture, CPU features, peripheral subsystems).
- Contrast the different types of processors for embedded systems: CPU microcontrollers, DSP processors, GPUs, heterogeneous SOCs (CPUs/accelerators), FPGA-based processors.

CE-ESY-4 Basic software techniques for embedded applications

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Manually translate simple high-level language statements to equivalent assembly language.
- Describe the actions of compilation, assembly, linking in the program translation process.
- Describe actions taken by compiler-generated code after system reset but before user application execution.
- Describe memory assignments made by a compiler for global variables, local variables, subroutine parameters and dynamically allocated storage.
- Explain the basic loop-forever structure of an embedded program.
- Design simple programs for embedded system applications including some that include modular/hierarchical programming techniques such as subroutines and functions.
- Demonstrate debugging techniques for simple embedded application programs.

CE-ESY-5 Parallel input and output
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Describe the appropriateness of different I/O configurations (input, strong drive, weak pullup/pulldown, open-drain, tri-state) available in general purpose I/O (GPIO) for a given target application.
- Create programs that perform a set of input/output operations on one more GPIOs using a polled approach.
- Describe how interrupts are supported on the target embedded system(s).
- Create programs that perform a sequence of input/output operations on one more GPIOs using an interrupt-driven approach.
- Discuss mechanisms such as hardware and software FIFOs for buffering data streams.

Elective Learning Outcomes:
- Discuss Direct Memory Access (DMA) and describe how it is supported on the target embedded system.
- Create programs that perform a sequence of input/output operations using DMA.

CE-ESY-6 Asynchronous and synchronous serial communication
Minimum core coverage time: 6 hours

Core Learning Outcomes:
- Discuss the concepts of full-duplex and half-duplex communication.
- Contrast parallel I/O versus serial I/O tradeoffs in terms of throughput, wiring cost, and application.
- Describe the data formatting, timing diagrams, and signaling levels used in an asynchronous serial interface.
- Create programs that perform I/O to an external device or system that uses an asynchronous serial interface.
- Describe the data formatting, timing diagrams, and signaling levels used in a synchronous serial interface such as SPI or I2C.
- Create programs that perform I/O to an external device or system that uses a synchronous serial interface such as SPI or I2C.

CE-ESY-7 Periodic interrupts, waveform generation, time measurement
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Describe the basic features and operation of typical hardware timers used in embedded systems.
- Create programs that perform periodic I/O triggered by hardware timer-generated interrupts.
- Create programs that measure waveform characteristics such as pulse width and frequency using hardware timers.
- Describe applications of pulse width modulation.
- Create programs that use pulse width modulation for external device control.

CE-ESY-8 Data acquisition, control, sensors, actuators
Minimum core coverage time: 4 hours

Core Learning Outcomes:
- Describe terms and properties relating to Analog-to-Digital Conversion (ADC) and Digital-to-Analog Conversion (DAC) such as sampling rate, reference voltage, conversion time, precision, range, and encoding method.
- Perform voltage to binary and binary to voltage numerical conversions given range, encoding method, and reference voltage parameters.
- Describe DAC and ADC architectural approaches such as resistor ladder, successive approximation, flash, and delta-sigma, and give tradeoffs such as conversion time and circuit complexity.
- Demonstrate numerical conversion from a physical quantity such as pressure, temperature, and acceleration to voltage or current given an example sensor and its characteristic equation or graph.
- Create programs that use one or more external sensors for monitoring physical properties.
- Demonstrate numerical conversion from voltage or current to a physical quantity such as linear/angular movement, sound, and light given an example actuator and its characteristic equation or lookup-up table.
- Create programs that use one or more actuators for effecting physical control by an embedded system.
- Design circuitry that transforms voltage level/current drive from/to external sensors/actuators to that required/provided by a target CPU.

CE-ESY-9 Implementation Strategies for Complex Embedded Systems
Minimum core coverage time: 7 hours

Core Learning Outcomes:
- Describe the need for structured approaches in writing complex embedded applications.
- Describe techniques used in event-driven state machine frameworks such as events, event queues, active objects, event processing,

priority queues, and hierarchical state machines.
- Describe techniques used in real time operating systems (RTOS) such as message passing, preemptive versus cooperative scheduling, semaphores, queues, tasks, co-routines, and mutexes.
- Create programs using either a state machine framework or an RTOS (or both) for sample embedded system applications.

CE-ESY-10 Techniques for low-power operation
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Describe sources of energy consumption such as toggling, leakage and pin configurations used for minimizing power.
- Describe power saving approaches used in embedded system design and their corresponding performance/power tradeoffs such as sleep/hibernate modes, peripheral system enable/disable, and clock frequency management, and appropriate GPIO configurations during sleep/hibernate.
- Describe wakeup mechanisms such as watchdog timer, real time clock, and external interrupts.
- Write programs that demonstrate minimal energy usage in performing I/O tasks through use of sleep and/or hibernate modes.
- Compute system battery life for an embedded system platform given parameters such as battery capacity, current draw, wake time, sleep time, clock frequency.

CE-ESY-11 Mobile and networked embedded systems
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Describe the role of embedded systems in the 'internet of things.'
- Discuss options for adding short-range wireless connectivity to an embedded system such as Bluetooth and 802.15.4 and tradeoffs relating to cost, power, throughput, and connectivity.
- Discuss options for adding long-range wireless connectivity to an embedded system such as cellular and Ethernet and tradeoffs relating to cost, power, throughput, and connectivity.
- Contrast hardware options for adding wireless connectivity to an embedded system such as external smart modules or software stack-plus-radio integrated circuits.
- Contrast connectivity architectures such as point-to-point, star, and mesh.
- Discuss security options for protecting wireless communication links.

CE-ESY-12 Advanced input/output issues
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Discuss concepts used in I/O buses such as master/slave devices, arbitration, transactions, priorities, and packets.
- Contrast single-ended signaling versus differential signaling for use in high-speed serial busses, and methods for measuring differential signaling quality such as eye-diagrams.
- Describe features such as topology, signaling levels, arbitration, speed, packet structure, and data transfers for one or more advanced serial bus protocols such as the Controller Area Network, Universal Serial Bus, and IEEE 1394 (FireWire).
- Discuss architectures and applications of persistent storage for embedded systems, such as flash drives, SD cards, and FRAM.

CE-ESY Supplementary Knowledge Units

CE-ESY-13 Computing platforms for Embedded Systems
Supplementary

Elective Learning Outcomes:
- Describe multimedia peripherals found in advanced embedded System-On-Chip implementations such video encoding, audio processing, display processing.
- Describe interconnect and networking options for SoCs, including Network-on-Chip architectures.
- Contrast performance, power, and flexibility tradeoffs for hard core versus software CPUs found in Field Programmable Gate Arrays.
- Describe embedded applications that benefit from a multi-core approach.
- Describe embedded applications that benefit from other types of processors for embedded systems: DSP processors, GPUs, heterogeneous SOCs (CPUs/accelerators), FPGA-based processors.

CE-NWK Computer Networks

[20 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. Development history of computer network, network hierarchy and the important role of computer network in the computer industry
2. Related standards and common tools (e.g., tools for performance evaluation and network topology) used in research of computer networks
3. Architecture of computer networks, the OSI model, and the TCP/IP model
4. Fundamentals and technologies in data communication and transfer protocols of the physical layer and the data link layer
5. LAN networking, protocols of the MAC layer, and concepts and features of WAN
6. The network layer, the transport layer, the application layer, and typical network applications, such as e-mail, www, and ftp
7. Tradeoffs associated with various network architectures and protocols
8. Basic concepts, purposes, and common protocols of network management
9. Features and networking technologies of wireless sensor networks

CE-NWK Core Knowledge Units

CE-NWK-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Describe the origins and development history of computer networks.
- Explain important applications of computer network.
- Identify people who made important contributions to networks and specify the contributions they made.
- Explain the basic composition and hierarchy of computer network.
- Discuss the role of hierarchy in computer network construction.
- Explain the main protocols and the key technologies related to computer networks.

CE-NWK-2 Relevant tools, standards and/or engineering constraints

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Describe the broad taxonomy of wireless standards such as cellular network standards vs. 802 family standards.
- Provide an overview of the IEEE 802 family standards including IEEE802.3, 802.11, 802.15, and 802.16.
- Provide an overview of cellular network standards including 2G, 3G, 3.5G, 4G, 5G, and LTE.
- Explain the Bluetooth wireless technology standard.
- Contrast functions and basic usages of a modern network simulator.
- Discuss constraints of the development of computer networks, such as transmission media, network security and network management.

CE-NWK-3 Network architecture

Minimum core coverage time: 4 hours

Core Learning Outcomes:

- State the fundamental concepts of networks and their topologies.
- Contrast network architectures with the network's hardware components.
- Contrast the elements of a protocol with the concept of layering.
- Explain the importance of networking standards and their regulatory committees.
- Describe the seven layers of the OSI model.
- Define the role of networking and internetworking devices such as repeaters, bridges, switches, routers, and gateways.
- Explains the pros and cons of network topologies such as mesh, star, tree, bus, ring, and 3-D torus.
- Describe the TCP/IP model.
- Contrast the TCP/IP model with the OSI model.

CE-NWK-4 Local and wide area networks

Minimum core coverage time: 4 hours

Core Learning Outcomes:

- Explain the basic concepts of LAN, MAN, and WAN technologies, topologies, and associated tradeoffs.

- Describe the use of network technologies for on-chip interconnect networks such as Network-on-Chip (NoC) architectures.
- Contrast different components and requirements of network protocols with their tradeoffs.
- Explain the functions of the physical layer and describe features of different transmission media and technologies.
- Articulate the basic concepts of error detection and correction at the data-link layer.
- Contrast circuit and packet switching
- Explain the access and control methods of common shared media.
- Contrast key innovations of Ethernet and Gigabit Ethernet.
- Explain the key concepts of carrier-sense multiple-access networks (CSMA).
- Explain how to build a simple network using a network protocol that operates at the physical and data-link layers of the OSI model.

Elective Learning Outcomes:

- Describe protocols for addressing and congestion control.
- Describe protocols for virtual circuits and quality of service.

CE-NWK-5 Wireless and mobile networks

Minimum core coverage time: 2 hours

Core Learning Outcomes:

- Explain the source of changes in the wireless and mobile industry from the view point of new service models such as the mobile ecosystem (e.g., Apple and Android ecosystems).
- Describe the fundamental components that tend to be unchanged for long periods such as mobile IP, Wi-Fi, and cellular.
- Explain the potential issues in wireless media access such as the hidden terminal problem and the exposed terminal problem.
- Explain the basics of a Wi-Fi network such as protocol stack and frame structure as well as its development such as IEEE802.11 a/b/g/n series standards.
- Contrast the basic concepts in cellular network such network architecture, framework, and LTE.
- Describe the main characteristics of mobile IP and explain how it differs from standard IP regarding mobility management and location management; illustrate how traffic is routed using mobile IP.
- Describe features of typical wireless MAC protocols.

Elective Learning Outcome:

- Explain wireless CSMA/CA and RTS/CTS enhancement mechanisms.

CE-NWK-6 Network protocols

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Contrast connection-oriented and connectionless services.
- Contrast network protocols in dimensions related to their syntax, semantics, and timing.
- Define the role of key layers from a software stack including physical-layer networking concepts, data-link layer concepts, internetworking, and routing.
- Explain some common protocol suites and the services they provide (e.g., IPv4, IPv6, and TCP/UDP).
- Describe the functions of the network layer and networking technology.
- Contrast different network architectures.
- Describe the important technologies used in routers.
- Contrast at least two important routing algorithms.
- Explain congestion control and contrast its related algorithms.
- Describe main contents of the IP, TCP and UDP protocols.
- Explain the role of the Domain Name System (DNS) and the benefits of its distributed design.

CE-NWK-7 Network applications

Minimum core coverage time: 2 hours

Core Learning Outcomes:

- Describe the key components of a web solution stack using LAMP (Linux, Apache HTTP server, MySQL, PHP/Perl/Python) or other similar illustrative examples.
- Explain the different roles and responsibilities of clients and servers for a range of possible applications.
- Select a range of tools that ensures an efficient approach to implementing various client-server possibilities.
- Design and build a simple interactive web-based application (for example, a simple web form that collects information from the client and stores it in a file on the server).
- Discuss web software stack technologies such as LAMP solution stack.
- Explain characteristics of web servers such as handling permissions, file management, and capabilities of common server architectures.
- Describe support tools for website creation and web management.
- Describe at a high level, ways in which a wide variety of clients and server software interoperates to provide e-mail services worldwide.

Elective Learning Outcomes:

- Implement solutions using dynamic HTML and client- and server-side models for web applications.
- Give examples of and state advantages and disadvantages of peer-to-peer models.
- Explain the principles, advantages, and challenges of cloud computing.
- Give examples of cloud computing APIs or commercial services and summarize the key abilities they provide.
- Describe the key components and tradeoffs of a modern network application that requires a hybrid of many areas with computer networks such as machine learning, data mining, HCI, and transportation systems.

CE-NWK-8 Network management

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Discuss the possible objectives and main instruments for network management.
- Describe the role of a domain name server (DNS) in distributed network management.
- Describe common network management protocols such as ICMP, and SNMP.
- Contrast three main issues related to network management.
- Discuss four typical architectures for network management including the management console, aggregators, and device agents.
- Demonstrate the management of a device such as an enterprise switch through a management console.
- Contrast various network management techniques as they apply to wired and wireless networks such as topics on devices, users, quality of service, deployment, and configuration of these technologies.
- Discuss the address resolution protocol (ARP) for associating IP addresses with MAC addresses.
- Explain two quality of service issues such as performance and failure recovery.
- Describe ad hoc networks.
- Explain troubleshooting principles and techniques related to networks.
- Describe management functional areas related to networks.

CE-NWK Supplementary Knowledge Units

CE-NWK-9 Data communications

Supplementary

Elective Learning Outcomes:

- Define the fundamental concepts of data communications.
- Apply signals and signal encoding methods to communication service methods and data-transmission modes.
- Explain the role of modulation in data communication.
- Contrast the issues involved with A/D and D/A conversion in data communications.
- Contrast communication hardware interfaces such as modems.
- Explain various approaches to multiplexing.
- Explain the basic theory of error detecting and correcting codes and provide an example.

CE-NWK-10 Performance evaluation

Supplementary

Elective Learning Outcomes:

- Describe performance metrics.
- Contrast how different performance metrics affect a specific network and/or service paradigm.
- Contrast service paradigms such as connection-oriented service and connectionless service.
- Contrast network performance characteristics including latency and throughput.
- Discuss network error sources such as dropped packets and corrupted data.
- Define a “quality of experience” metric (QoE, QoX or simply QX), which is a measure of a customer's experiences with a service (e.g., web browsing, phone call, TV broadcast, call to a call center).
- Apply fundamental modeling theory to analyze the performance of a network (e.g., a M/M/1 queue).

CE-NWK-11 Wireless sensor network

Supplementary

Elective Learning Outcomes:

- Describe the features of wireless sensor network (WSN) systems.
- Describe the MAC and routing protocols of WSNs.
- Discuss the requirements and strategies of WSN data fusion.
- Provide an example of a real application of WSNs.
- Contrast circuit switching vs packet switching: virtual circuit switching (MPLS).

CE-PPP Preparation for Professional Practice

[20 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. The importance of effective communication among professionals and other diverse audiences
2. The significance of leadership and professional interaction when functioning within an interdisciplinary team
3. The professional and ethical responsibilities of practicing computer engineers and the effects of their work on society
4. The importance of understanding contemporary issues, lifelong learning strategies, and legal and intellectual property issues
5. The importance of business acumen and skill in managing projects in the computer engineering field

CE-PPP Core Knowledge Areas

CE-PPP-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Describe the nature of professionalism and its place in the field of computer engineering.
- Identify some contributors and relate their achievements to social and professional issues.
- Contrast ethical and legal issues as related to computer engineering.
- Indicate reasons for studying social and professional issues.
- Identify stakeholders in an issue and an engineer's obligations to them.
- Explain professionalism and licensure relative to a practicing computer engineer.
- Describe how computer engineering uses or benefits from social and professional issues.

CE-PPP2 Relevant tools, standards, and/or engineering constraints

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Interpret the social context of a specific implementation.
- Identify non-technical assumptions and values that an engineer would associate with the design of a computer component.
- Explain why "freedom of expression" in cyberspace is important in computer engineering.
- Describe positive and negative ways in which computer engineering alters the modes of interaction between people.
- Explain why computing/network access is restricted in some countries.
- Illustrate the use of example, analogy, and counter-analogy in an ethical argument.
- Contrast what is legal with what is ethical.
- Explain the importance of ethical integrity in the practice of computer engineering.

CE-PPP-3 Effective communication strategies

Minimum core coverage time: 2 hours

Core Learning Outcomes:

- Listen attentively in technical and non-technical contexts.
- Write technical reports using correct spelling and grammar.
- Use compelling arguments when writing technical reports and when making oral presentations.
- Become an assertive communicator in writing and in speaking.
- Build positive rapport with an audience.
- Develop strategies for effective communication in writing and in speaking.
- Describe ways in which body language affects communication.
- Use appropriate visual aids for effective communication.
- Use visualization skill in presenting a technical paper or report.
- Write technical reports per specified guidelines.
- Use a presentation mode appropriate for a wide range of audiences.
- Use a writing style appropriate to a wide range of audiences.
- Engage with an audience in response to questions.
- Write technical reports that are well organized and structured per accepted standards.

CE-PPP-4 Interdisciplinary team approaches

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Describe the meaning of interdisciplinary teams in two different contexts.
- Develop a possible skill set needed to function effectively on an interdisciplinary team.
- Describe some computer engineering projects where interdisciplinary approaches are important.
- Explore ways in which industry approaches teamwork toward a common goal.
- Create an interdisciplinary team for a given project by assigning roles and responsibilities for each team member.
- Identify situations that would undermine interactions among members of an interdisciplinary team.
- Explore ways in which one might assess the performance of an interdisciplinary team.
- Describe possible assessment methods used to monitor interdisciplinary teams.

CE-PPP-5 Philosophical frameworks and cultural issues

Minimum core coverage time: 2 hours

Core Learning Outcomes:

- Summarize the basic concepts of relativism, utilitarianism, and deontological theories.
- Describe some engineering problems related to ethical relativism.
- Describe the differences between scientific and philosophical approaches to computer engineering dilemmas.
- Contrast the distinction between ethical theory and professional ethics.
- Identify the weaknesses of the "hired agent" approach, strict legalism, naïve egoism, and naïve relativism as ethical frameworks.
- Contrast Western and non-Western philosophical approaches and thought processes as they apply to the computer engineering field.

CE-PPP-6 Engineering solutions and societal effects

Minimum core coverage time: 2 hours

Core Learning Outcomes:

- Articulate the importance of product safety when designing computer systems.
- Describe the differences between correctness, reliability, and safety.
- Explain the limitations of testing to ensure correctness.
- Describe other societal effects beyond risk, safety, and reliability.
- Identify unwarranted assumptions of statistical independence of errors.
- Discuss the potential for hidden problems in reuse of existing components.
- Explain ways computer engineers would assess and manage risk, and how they would inform the public of risk.
- Articulate ways public perception of risks often differs from actual risk, as well as the implications of this difference.
- Explain why product safety and public consumption should be a hallmark of computer engineering.

CE-PPP-7 Professional and ethical responsibilities

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Identify progressive stages in a whistle-blowing incident.
- Specify the strengths and weaknesses of relevant professional codes as expressions of professionalism.
- Identify ways professional codes could become guides to decision making.
- Explore some historical examples of software risks such as the Therac-25 case.
- Provide arguments for and against licensure in computer engineering.
- Provide arguments for and against licensure in non-engineering professions.
- Identify ethical issues that may arise in software development and determine how to address them technically and ethically.
- Develop a computer use policy with enforcement measures.

CE-PPP-8 Intellectual property and legal issues

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Describe the foundations of intellectual property.
- Distinguish among patent, copyright, and trade secret protection.
- Contrast between a patent and a copyright.
- Outline some of the transnational issues concerning intellectual property.
- Discuss the legal background of copyright in national and international law.
- Explain ways patent and copyright laws might vary internationally.
- Outline the historical development of software patents and Contrast with other forms of intellectual property protection for software.
- Distinguish among employees, contractors, and consultants and the implications of each group.
- Explore a patent related to computer engineering and provide a summary of its content.
- Explain product and professional liability and articulate their applicability within computer engineering.
- Analyze the ethical ramifications of free open-source hardware and software.

CE-PPP-9 Contemporary issues
Minimum core coverage time: 2 hours

Core Learning Outcomes:
- Summarize the legal bases for the right to privacy and freedom of expression in one's own country.
- Discuss ways in which privacy varies from country to country.
- Describe current computer-based threats to privacy.
- Contrast the difference between viruses, worms, and Trojan horses.
- Explain how the internet might change the historical balance in protecting freedom of expression.
- Articulate some of the privacy implications related to massive database systems.
- Outline the technical basis of viruses and denial-of-service attacks.
- Provide examples of computer crime and articulate some crime prevention strategies.
- Define cracking and contrast it to hacking in computer engineering.
- Enumerate techniques to combat "cracker" attacks.
- Discuss several "cracker" approaches and motivations.

CE-PPP-10 Business and management issues
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Assess the total job cost of a project.
- Apply engineering economic principles when considering fiscal arrangements.
- Summarize the rationale for antimonopoly efforts.
- Describe several ways in which shortages in the labor supply affect the information technology industry.
- Explain ways in which computer engineers should cost out jobs with considerations of manufacturing, hardware, software, and engineering implications.
- Contrast some of the prospects and pitfalls in entrepreneurship.
- Describe the economic implications of monopolies.
- Describe the effects of skilled labor supply and demand concerning the quality of computing products.
- Summarize the rationale for antimonopoly efforts.
- Contrast two pricing strategies one might use in the computing domain.

CE-PPP Supplementary Knowledge Units

CE-PPP-11 Tradeoffs in professional practice
Supplementary

Elective Learning Outcomes:
- Indicate some important tradeoffs a computer engineer may have to make while practicing professionally.
- Articulate some ethical tradeoffs when making technical decisions.
- Describe some unethical tradeoffs that might occur in professional practice.
- Evaluate the risks of entering one's own business.
- Identify the professional's role in security and the tradeoffs involved.
- Describe the tradeoff between security and privacy, particularly as reflected in the post-9-11 era.
- Defend ways to address limitations on access to computing.

CE-SEC Information Security

[20 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. Recognition that security is risk management and inherently includes tradeoffs
2. Familiarity with the implications of hostile users, including social engineering attacks and misuse cases
3. Framework for understanding algorithms and other technological measures for enhancing security
4. Strategic and tactical design issues in information security

CE-SEC Core Knowledge Units

CE-SEC-1 History and overview

Minimum core coverage time: 2 hours

Core Learning Outcomes:

- State examples of famous security breaches and denials of service.
- Discuss common computer crime cost estimates and the difficulty of estimating them.
- Define ethical hacking.
- Contrast active with passive attacks.
- Discuss the issues surrounding computer security and privacy rights.
- Enumerate various motivations of attackers.
- Identify the types and targets of computer crime.
- Summarize the major types of attacks performed by cybercriminals.
- Discuss the professional's role in security and the tradeoffs involved.
- Give examples of historic and contemporary cryptography algorithms.
- Justify the use of various security principles (e.g. defense in depth, functional vs. assurance requirements, security through obscurity is flawed, security is risk management, complexity is the enemy of security, and benefits of responsible open disclosure).
- Explain and defend the use of each of various security mechanisms (e.g., least privilege, fail-safe defaults, complete mediation, separation of privilege, and psychological acceptability).

CE-SEC-2 Relevant tools, standards, and/or engineering constraints

Minimum core coverage time: 2 hours

Core Learning Outcomes:

- Discuss the major provisions of a relevant law such as HIPAA or the EU Data Protection Directive.
- Summarize intellectual property and export control laws affecting security, especially encryption.
- Describe some common approaches and tools used in penetration testing.
- Articulate some challenges of computer forensics.

CE-SEC-3 Data security and integrity

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Define confidentiality and integrity.
- Give examples of systems where integrity alone is sufficient.
- Define "perfect forward secrecy" and explain why it is desirable.

CE-SEC-4 Vulnerabilities: technical and human factors

Minimum core coverage time: 4 hours

Core Learning Outcomes:

- Define misuse cases and explain their role in information security.
- Describe the role of human behavior in security system design, including examples of social engineering attacks.
- Perform a simple fault tree analysis.
- Explain the types of errors that fuzz testing can reveal.
- Discuss issues related to the difficulty of updating deployed systems.
- Explain the role of code reviews in system security.
- Define the problem of insecure defaults.
- Explain the tradeoffs inherent in responsible disclosure.

- Discuss why the advantage is with the attacker in many contexts and how this must be addressed in system design.
- Explain how to execute a stack overrun attack and the knowledge it requires.
- Discuss recent examples of exploited memory access bugs and the errors that lead to their deployment and exploitation.
- Explain the role of both safe libraries and argument validation in defending against buffer overflows.
- Illustrate how a stack canary works.
- Explain the problems solved by address space randomization and non-executable memory.
- Define several types of malware such as viruses, worms, Trojan horses, key loggers, and ransomware.
- Discuss countermeasures to common types of malware.
- Explain current issues in the "arms race" between malware authors and defense system authors.

CE-SEC-5 Resource protection models
Minimum core coverage time: 1 hour

Core Learning Outcomes:
- Explain the pros and cons of various discretionary and mandatory resource protection models.
- Illustrate an access control matrix model.
- Define the Bell-LaPadula model.

CE-SEC-6 Secret and public key cryptography
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- State the motivation for putting all encryption algorithm variability in the keys.
- Discuss the effect of processing power on the effectiveness of cryptography.
- Explain the meaning of and relationship between the three basic classes of cryptographic attacks: ciphertext only, known plaintext, chosen plaintext.
- Discuss the similarities and differences among the three basic types of cryptographic functions (zero-, one-, and two-key): hash, secret key, and public key.
- Discuss block and key length issues related to secret key cryptography.
- Describe and evaluate a symmetric algorithm such as advanced encryption standard (AES), focusing on both design and implementation issues.
- Explain some uses of one-time pads.
- Perform modular arithmetic (addition, multiplication, and exponentiation).
- Apply the basic theory of modular arithmetic (Totient function and Euler's theorem).
- Execute and apply the RSA algorithm for encryption and digital signatures.
- Execute and apply the Diffie-Hellman algorithm for establishing a shared secret.
- Demonstrate and discuss the motivations and weaknesses in various methods for applying secret key (block) encryption to a message stream such as cipher block chaining (CBC), cipher feedback mode (CFB), and counter mode (CTR).

CE-SEC-7 Message authentication codes
Minimum core coverage time: 1 hour

Core Learning Outcomes:
- Explain why hashes need to be roughly twice the length of secret keys using the birthday problem.
- Discuss the uses of hashes for fingerprinting and signing.
- Discuss the key properties of a cryptographic hash function contrasted with a general hash function.
- Explain the purpose of key operations used in cryptographic hashes such as permutation and substitution.
- Explain how one can use a hash for a message authentication code (MAC).
- State key properties of secure hash algorithms or family such as SHA-3, or its successor.
- Explain the problem solved by the HMAC standard.

CE-SEC-8 Network and web security
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Describe the goals of Transport layer security (TLS) and how they are attained using secret and public key methods along with certificates
- Discuss the reasons for using a firewall, various topologies, and firewall limitations.
- Diagram and explain the use of virtual private networks (VPNs).
- Describe common denial of service attack methods, including distributed and amplified attacks, along with countermeasures taken at the computing system, protocol design, and backbone provider levels.
- Define packet filtering.
- Discuss the ramifications of the HTTP/HTTPS web platform design being stateless.

- Describe the basic structure of URLs, HTTP requests, and HTTP digest authentication as they relate to security.
- Explain the use of HTTP cookies including session cookies, expiration, and re-authentication for key operations.
- Define cross-site scripting.
- Explain an SQL injection attack and various methods of remediation.
- Be familiar standards such as open web application security project (OWASP) and the OWASP Top 10 list.

CE-SEC-9 Authentication
Minimum core coverage time: 1 hour

Core Learning Outcomes:
- Explain the difference between authorization and authentication.
- Comment on authentication methods using password and/or address-based methods.
- Discuss eavesdropping and server database reading and explain how various authentication methods deal with them.
- Explain the general use of trusted intermediaries for both secret and private key systems.
- Discuss issues specific to authenticating people, including the three main approaches to doing so.
- Describe the problems solved by multi-factor authentication methods including biometrics.

CE-SEC-10 Trusted computing
Minimum core coverage time: 1 hour

Core Learning Outcomes:
- Describe current approaches to trusted computing such as trusted hardware, secure storage, and biometrics.
- Evaluate a circumvention method for a trusted computing system and discuss the tradeoffs between implementation cost, information value, and circumvention difficulty.

CE-SEC-11 Side-channel attacks
Minimum core coverage time: 1 hour

Core Learning Outcomes:
- Discuss various side channels and methods of encoding information on them.
- Discuss the tradeoffs of side-channel protection and system usability.

CE-SGP Signal Processing

[30 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. Need for and tradeoffs made when sampling and quantizing a signal
2. Linear, time-invariant system properties
3. Frequency as an analysis domain complementary to time
4. Filter design and implementation
5. Control system properties and applications

CE-SGP Core Knowledge Units

CE-SGP-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Explain the purpose and role of digital signal processing and multimedia in computer engineering.
- Explain some important signal processing areas such as digital audio, multimedia, image processing, video, signal compression, signal detection, and digital filters.
- Contrast analog and digital signals using the concepts of sampling and quantization.
- Draw a digital signal processing block diagram and define its key components: antialiasing filter, analog to digital converter, digital signal processing, digital to analog filter, and reconstruction filter.
- Explain the need for using transforms and how they differ for analog and discrete-time signals.
- Contrast some techniques used in transformations such as Laplace, Fourier, and wavelet transforms.
- Indicate design criteria for low- and high-pass filters.

CE-SGP-2 Relevant tools, standards, and/or engineering constraints

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Describe the tradeoffs involved with increasing the sampling rate.
- Indicate key issues involved with sampling periodic signals including the sampling period.
- Indicate key issues involved with sampling non-periodic signals including spectral resolution.
- Prove whether a system is linear, time-invariant, causal, and/or stable given its input to output mapping.
- Derive non-recursive and recursive difference equations, as appropriate, given descriptions of input-output behavior for a linear, time-invariant system.

CE-SGP-3 Convolution

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Explain how the concept of impulse response arises from the combination of linearity and time-invariance.
- Derive the linear convolution summation from the definition of impulse response and linearity.
- Use the commutative property of convolution as a foundation for providing two explanations of how a system output depends on the input and system impulse response.

CE-SGP-4 Transform analysis

Minimum core coverage time: 5 hours

Core Learning Outcomes:

- State, prove, and apply properties of the z–transform and its inverse.
- State, prove, and apply properties of the discrete-time Fourier transform (DTFT) and its inverse.
  - Explain how the DTFT may be interpreted as a spectrum.
  - Explain the relationship between the original and transformed domains (e.g., aliasing).
- State, prove, and apply properties of discrete Fourier transform (DFT) and its inverse.
- Prove and state the symmetries of the Fourier transforms for real signals.
- State the frequency shift property for Fourier transforms.
- Prove and state how Parseval's theorem relates power or energy, as appropriate, for the Fourier transforms.
- Explain the relationship among the z-transform, DTFT, DFT, and FFTs (fast Fourier transforms).

- Define and calculate the Laplace transform of a continuous signal.
- Define and calculate the inverse Laplace transform.

CE-SGP-5 Frequency response
Minimum core coverage time: 5 hours

Core Learning Outcomes:
- Interpret the frequency response of an LTI system as an alternative view from the impulse response.
- Analyze the frequency response of a system using the DTFT and the DFT.
- Determine pole and zero locations in the z-plane given a difference equation describing a system.
- Relate the frequency selectivity of filters to the z–transform domain system representation.
- Describe the repeated time series implication of frequency sampling.

CE-SGP-6 Sampling and aliasing
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- State the sampling theorem and the related concepts of the Nyquist frequency and aliasing.
- Demonstrate aliasing on a sampled sine wave.
- State the relationship between time and frequency domains with respect to sampling.
- Explain when spectra are discrete vs. continuous.
- Calculate the errors or noise generated by sampling and quantizing.

CE-SGP-7 Digital spectra and discrete transforms
Minimum core coverage time: 6 hours

Core Learning Outcomes:
- Sketch the spectrum of a periodic signal.
- Contrast the spectra of an impulse and a square wave.
- Calculate spectra of periodic and aperiodic signals.
- Explain how the block size controls the tradeoff between spectral resolution and density.
- Calculate a spectrogram and explain what its key parameters are.
- Explain filtering as adding spectra in a frequency domain on a logarithmic scale.
- Design interpolation and reconstruction filters using the sinc function.

CE-SGP-8 Finite and infinite impulse response filter design
Minimum core coverage time: 4 hours

Core Learning Outcomes:
- Design finite and infinite impulse response (FIR and IIR) filters that have specified frequency characteristics including magnitude and phase responses.
- Explain the general tradeoffs between FIR and IIR filters.
- Demonstrate that not all recursive filters are IIR, using a moving average as an example.
- Use the DFT to accomplish filtering through (circular) convolution.
- State the condition for linear phase in an FIR filter.
- Explain the tradeoffs between spectral resolution, length, and delay in an FIR filter.
- Explain why one or more FIR filter design methods work.
- Explain why one or more IIR filter design methods work including notch filters using pole-zero pairs.
- Design a digital filter using analog techniques (e.g., bilinear transform) and explain its key parameters.
- Explain physically realizable system issues relevant in filter design including causality and time shifts, and response truncation.

CE-SGP Supplementary Knowledge Units

CE-SGP-9 Window functions
Supplementary

Elective Learning Outcomes:
- Explain how window functions improve transform properties.
- Explain the periodic assumption in spectral analysis.
- Explain the purpose of a window function and its effect on a spectrum.
- Discuss the tradeoffs of common window functions such as rectangular, Blackman, Hann, and Hamming.

- Select an appropriate window function given a problem statement regarding detection or identification tradeoffs.

CE-SGP-10 Multimedia processing
Supplementary

Elective Learning Outcomes:
- Define signals that vary in time and/or space and interpret frequencies in both domains.
- Describe how sampling affects image integrity.
- Explain how low-pass filtering tends to smooth images.
- Contrast between reconstruction and enhancement filters.
- Describe methods for minimizing image noise.
- Describe how digital techniques perceptually or otherwise enhance speech and audio signals.
- Explain techniques for noise reduction (e.g., Weiner or median filters) or cancellation (e.g., LMS filters) in audio processing.
- Explain the motivation for audio coding and state key elements of MPEG or related algorithms including perceptual elements.

CE-SGP-11 Control system theory and applications
Supplementary

Elective Learning Outcomes:
- Define basic control system concepts (e.g., zero-state response, zero-input response, stability).
- Contrast design methods (root-locus, frequency-response, state-space) for control systems.
- Explain limitations and trade-offs associated with microcontroller implementations of digital control systems.
- Describe potential applications of digital control systems for electro-mechanical systems, including robotics.
- Implement a simple microcontroller-based motion control system with sensors and actuators.

CE-SPE Systems and Project Engineering

[35 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. The role of systems engineering principles throughout a computer system's life cycle, including important tradeoffs in such areas as power, performance, and cost
2. Project management, including team management, scheduling, project configuration, information management, and design of project plans
3. Human-computer interaction styles and usability requirements, design of user interfaces, and input/output technologies
4. Analysis and design to produce desired levels of risk, dependability, safety, and fault tolerance in computer-based systems
5. System requirements and methods for eliciting and analyzing requirements for a computer-based system
6. System specifications, their relationship to requirements and system design, and methods for developing and evaluating quality specifications for computer-based systems
7. System architectural design and evaluation, including tools and methods for modeling, simulating, and evaluating system designs at the architectural level
8. Methods and tools for concurrent hardware and software design, system integration, testing, and validation, including unit and system level test plans
9. Design for manufacturability, sustainability, and maintainability throughout the product life cycle

CE-SPE Core Knowledge Units

CE-SPE-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Articulate differences between software and hardware engineering, and computer systems engineering.
- Explain briefly the concept of a system and a subsystem, and discuss the role of people, the different disciplines involved, and the need for interdisciplinary approaches to the development of the range of computer-based systems.
- Indicate some important elements of computer systems engineering such as design processes, requirements, specifications, design, testing, validation, evolution, project management, hardware-software interface, and the human-computer interface.
- Define and explain product life cycle, the role of system engineering throughout a product life cycle, and reasons why many computer-based system designs become continually evolving systems.
- Provide reasons for the importance of testing, validation, and maintenance in computer systems development.
- Explain the importance of design decisions and tradeoffs at the systems level, including balancing costs, performance, power, dependability, and market considerations.

CE-SPE-2 Relevant tools, standards and/or engineering constraints

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Select, with justification, an appropriate set of tools to support the development of a range of computer-based systems, including tools for project management, requirements and specifications definition and analysis, configuration management, tradeoff analysis, and computer-aided tools for software, hardware, and systems design, including modeling, simulation, evaluation, and testing.
- Analyze and evaluate a set of tools in an area of computer system development (e.g., management, modeling, or testing).
- Demonstrate the ability to use a range of tools to support the development of a computer-based system of medium size. (This could be done in the context of a class project or assignment.)
- Explain the importance and influence of standards, guidelines, legislation, regulations, and professional issues on the development of computer-based systems.
- Describe tradeoffs that occur in following regulatory standards and regulations.

CE-SPE-3 Project management

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Describe basic elements of project management that support development of computer-based systems for a variety of applications, including interdisciplinary issues.
- Describe the different phases of a system's life cycle and identify tools to support these phases, including such project-management tools as Gantt charts for project planning, scheduling, cost analysis, resource allocation, and teamwork.
- Demonstrate, through involvement in team projects, the central elements of team building and team management, including team composition and organization, roles and responsibilities in a design team, decision-making processes, project tracking, and team

problem resolution.
- Describe methods and tools for project configuration management and management of project information, ensuring timely compliance with specifications and timely delivery.
- Prepare a project plan for a computer-system design project that includes estimates of size and effort, a schedule, resource allocation, configuration control, change management, and project risk identification and management—this could be done in the context of a class project or assignment.

Elective Learning Outcomes:
- Identify and describe the use of metrics in support of project management.
- Describe the roles of consultants and subcontractors in design projects, including their use and their management.
- Discuss how standards and legal requirements can affect the management of design projects.

CE-SPE-4 User Experience[2]
Minimum core coverage time: 6 hours

Core Learning Outcomes:
- Define the meaning of user experience (UX) and describe the evolution from human factors to user experience design (UXD).
- Contrast the physical and non-physical aspects of UXD.
- Summarize some common human-computer interaction styles, and discuss how one would analyze human interaction with computer-based systems.
- Describe common usability guidelines and standards; give examples of functional and usability requirements that can be beneficial in developing human-centered computer systems, including users with different abilities (e.g., age, physical disabilities).
- Identify fundamental principles for effective GUI design, relevant to different applications and different system platforms in computer engineering.
- Discuss tradeoffs involved when developing a UX system environment.
- Identify system components that are suitable for the realization of high-quality multimedia interfaces.
- Evaluate an existing interactive system with appropriate human-centered criteria and usability, giving reasons for selection of criteria and techniques.
- Discuss the role of visualization technologies in human-computer interaction.
- Explain the importance of social psychology in the design of user interfaces.
- Describe two main principles for universal design.
- List advantages and disadvantages of biometric access control.
- Describe a possible interface that allows a user with severe physical disabilities to use a website.
- Design, prototype, and conduct a usability test of a simple 2D GUI, using a provided GUI-builder, and, in doing so, create an appropriate usability test plan.

Elective Learning Outcomes:
- Discuss other techniques for interaction, such as command line interface and shell scripts.
- Identify the potential for the use of intelligent systems in a range of computer-based applications, and describe situations in which intelligent systems may, or may not, be reliable enough to deliver a required response.

CE-SPE-5 Risk, dependability, safety, and fault tolerance
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Recognize risk, dependability, and safety requirements for a range of computer-based systems, and discuss potential tradeoffs between these and other system requirements, such as performance and low-power operation.
- Explain the concepts of reliability, availability, and maintainability, as measures of system dependability, and explain their relationship to faults.
- Perform a risk analysis of a medium-size computer-based system.
- Identify at least two tradeoff concerns when developing a safety critical system.
- Indicate why it is important to know how to build dependable systems from unreliable components.
- Demonstrate an ability to model reliability, availability, and maintainability of simple computer-based systems.
- Describe some strategies for achieving desired levels of dependability, safety, and security.
- Discuss the nature of hardware and software faults, and redundancy methods used to tolerate them.
- Describe fault tolerance and dependability requirements of different applications (such as database, aerospace, telecommunications, industrial control, and transaction processing).

Elective learning outcomes:
- Describe one or more strategies for risk reduction and risk control, including implications for implementation.
- Discuss how international standards, legal requirements, and regulations relate to risk, safety and dependability impact the design of computer-based systems.

[2] User experience (UX) was formerly known as human-computer interaction (HCI)

- Discuss the use of failure modes and effects analysis (FEMA) and fault tree analysis in the design of high-integrity systems.
- Identify some hardware redundancy approaches for fault-tolerant system design, including the use of error detecting and correcting codes.
- Discuss one or more software approaches to tolerating hardware faults.
- Describe one or more methods for tolerating software faults, such as N-version programming, recovery blocks, and rollback and recovery.

CE-SPE-6 Hardware and software processes
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Explain the need for a disciplined approach to system development and the elements of a disciplined approach in specific contexts.
- Describe the nature of a life cycle, the role of life cycle models, quality in relation to the life cycle, the influence of system nature, and the size on choice of life cycle model.
- Describe some common software and hardware development models and show how to use these models during the development of a computer-based system.
- Explain how to gather data to inform, assess, and improve system design processes.
- Describe the benefits of agile methods for hardware and software design.
- Discuss the importance of modular design processes, and the design for modularity and reuse in the development of a computer-based system.
- Select, with justification, system development models most appropriate for the development and maintenance of diverse computer-based systems.

Elective Learning Outcomes:
- Explain the role of process maturity models, standards, and guidelines.
- Identify several metrics for software, hardware, and system processes.

CE-SPE-7 Requirements analysis and elicitation
Minimum core coverage time: 2 hours

Core Learning Outcomes:
- Perform an analysis of a proposed computer-based system design project, including identification of need, information gathering, problem definition, feasibility considerations, and economic considerations.
- Articulate a range of functional and non-functional requirements that might be applicable to the design of computer-based systems for a range of applications, and discuss how requirements can change as a system design project evolves.
- Discuss how tradeoffs between different system requirements might be necessary for a proposed computer-based system design.
- Describe the strengths and weaknesses of different approaches to requirements elicitation and capture.
- Apply one or more techniques for elicitation and analysis to produce a set of requirements for a medium-size computer-based system.
- Describe some quality factors for measuring the ability of a system design to meet requirements
- Conduct a review of a computer-based system requirements document using best practices to determine the document's quality.

Elective Learning Outcomes:
- Use a common, non-formal method to model and state—in the form of a requirements specification document—the requirements for a medium-size computer-based system (e.g., structured analysis or object-oriented-analysis).

CE-SPE-8 System specifications
Minimum core coverage time: 2 hours

Core Learning Outcomes:
- Discuss the relationship and differences between system specifications and requirements.
- Articulate some typical functional and nonfunctional specifications for the design of a computer-based system and the importance of specifications to the design process.
- Discuss one or more approaches for deriving system specifications from a requirements document.
- Discuss how tradeoffs between different system specifications might be necessary to meet system requirements.
- Assess the quality of a given specification, considering such factors as completeness, consistency, simplicity, verifiability, basis for design, specification in the event of failure, and degraded modes of operation.
- Given a set of requirements, create a high-quality specification for a computer-based system of medium complexity.
- Create a test plan, based on the specification, considering the role of independence in relation to test, safety cases, and limitations of such tests.

Elective Learning Outcomes:
- Describe and demonstrate the use of one or more formal specification languages and techniques.
- Translate into natural language a system specification written in a commonly used formal specification language.

CE-SPE-9 System architectural design and evaluation
Minimum core coverage time: 4 hours

Core Learning Outcomes:
- Describe concepts and principles of system architecture design, such as top-down design, subdivision into systems and subsystems, modularity and reuse, the hardware/software interface, and tradeoffs between various design options.
- Describe strengths and weaknesses of various systems-level architectural design methods, including procedural and functional methods.
- Describe design methods to meet system specifications and achieve performance measures, including dependability and safety.
- Given a system specification, select an appropriate design methodology (e.g., structured design or modular design) and create an architectural design for a medium-size computer-based system.
- Demonstrate ability to model, simulate, and prototype a range of computer-based system architectures.
- Using appropriate guidelines, conduct the review of one or more computer-based system designs to evaluate design quality based on key design principles and concepts.

Elective Learning Outcomes:
- At the architectural level, discuss possible failure modes, common cause failures, dealing with failure, inclusion of diagnostics in the event of failure, and approaches to fault-tolerant design.
- Discuss design issues associated with achieving dependability, the role of redundancy, independence of designs, separation of concerns, and specifications of subsystems.

CE-SPE-10 Concurrent hardware and software design
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Recognize the potential of hardware-software co-design in circumstances in which this approach is pertinent.
- Discuss how particular design constraints can make the coordinated development of both hardware and software important, such as in the design of low-power systems, real-time systems, or systems with high-performance requirements.
- Apply hardware-software co-design principles in situations of modest complexity.
- Discuss challenges to effective hardware-software co-design, such as demands of hard real-time features.
- Demonstrate ability to co-design to achieve specific technical objectives, such as low power, real-time operation, and high performance.
- Select and apply computer-aided tools to support hardware and software co-design.

CE-SPE-11 System integration, testing and validation
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Recognize the range of testing and validation methods appropriate for each stage of the system life cycle, including review of hardware models and software code; white box, black box, and regression testing; stress testing; and interface testing.
- Describe the role of various system validation tools and show how tools can support efficient and effective development.
- Discuss approaches to testing and validation at the unit level and at the integration and system levels.
- Create a test plan and generate test cases for a computer-based system of medium complexity, selecting an appropriate combination of tests for ensuring system quality.
- Demonstrate the application of the different types and levels of testing (unit, integration, systems, and acceptance) on computer-based systems of medium size.
- Undertake, as part of a team activity, an inspection of a medium-size computer-based system design.
- Discuss methods used for manufacturing test and inspection, and acceptance testing.

Elective Learning Objectives:
- Discuss methods for specialized testing: security, dependability/fault tolerance, and usability.

CE-SPE-12 Maintainability, sustainability, manufacturability
Minimum core coverage time: 2 hours

Core Learning Outcomes:
- Describe the need for, and characteristics of, maintainable software, hardware, and system designs.
- Discuss the inevitability of maintenance in certain systems, such as diagnosis, defect removal, hardware and/or software upgrades, and enhancement.
- Describe how to apply principles of maintainable design to a computer-based system of modest complexity.
- Identify issues associated with system evolution and explain their impact on the system life cycle.
- Explain configuration management and version control in engineering systems—the need for it, the issues associated with it, the

nature of the information to be held, legal requirements, and planning for possible disasters.
- Develop a plan for reengineering a medium-size product in response to a change request.
- Identify and exploit opportunities for component reuse in a variety of contexts.
- Discuss how design decisions can affect future generations, including impact on the environment and energy resources, and disposal of systems and components at end of life.
- Discuss design for manufacturability, part selection and standardization, manufacturing cost, and product lead-time for delivery.

CE-SRM System Resource Management

[20 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. Management of resources in computing systems with diverse components
2. Real-time operating constraints and their effect on system resource management
3. Resource management in mobile environments
4. Tradeoffs associated with resource management in different operating environments

CE-SRM Core Knowledge Units

CE-SRM-1 History and overview of operating systems

Minimum core coverage time: 1 hour

Core Learning Outcomes

- Explain the purpose of an operating system and the services one provides.
- Describe differences in functionality found in mobile, networked, client-server, distributed operating systems, and single user systems.
- Define key design criteria including efficiency, robustness, and security.
- Explain major threats to operating systems and how to guard against them.

CE-SRM-2 Relevant tools, standards, and/or engineering constraints

Minimum core coverage time: 1 hour

Core Learning Outcomes

- Give examples of real-time performance monitoring tools and log-based performance monitoring tools.
- Explain what information a real-time performance monitoring tool provides and when such a tool is useful.
- Explain what information a log-based performance monitoring tool provides and when such a tool is useful.
- List key components of the IEEE POSIX (Portable Operating System Interface) standard.
- Define the role of some key SRM APIs such as WinAPI and various Java APIs.

CE-SRM-3 Managing system resources

Minimum core coverage time: 8 hours

Core Learning Outcomes

- Describe the role of an operating system in managing system resources and interfacing between hardware and software elements.
- Explain what concurrency is and why it must be supported in managing system resources.
- Give examples of runtime problems that can arise due to concurrent operation of multiple tasks or components in the system, such as deadlock and race conditions.
- Describe basic types of interrupts and what must be done to handle them.
- Give examples that illustrate why task scheduling and dispatch are needed as system resources are managed.
- Explain the difference between preemptive and non-preemptive scheduling and demonstrate awareness of common algorithms used for scheduling.
- Describe how interrupts, dispatching tasks, and context switching are used to support concurrency.
- Explain the memory hierarchy (cache through virtual memory) and the cost-performance tradeoffs made in design.
- Describe the choices to be made in file system design and how these choices affect system resource management.

CE-SRM-4 Real-time operating system design

Minimum core coverage time: 4 hours

Core Learning Outcomes

- Explain the characteristics of hard real-time, soft real-time, and safety-critical real-time environments.
- Discuss issues of uncertainty that can arise in memory hierarchy design and disk or other fixed storage design and methods of addressing them.
- Explain latency and its role in RTOS design.
- Explain how an event-driven scheduler operates and be able give examples of commonly used algorithms.
- Explain how round-robin scheduling differs from event driven strategies.
- Explain why memory allocation is critical in a real-time system.
- Explain failure modes and recovery strategies appropriate for RTOS.
- Discuss tradeoffs among various operating system options.

CE-SRM-5 Operating systems for mobile devices
Minimum core coverage time: 3 hours

Core Learning Outcomes
- Describe the system communication that must be managed in a mobile device (e.g., Wi-Fi, Bluetooth).
- Discuss constraints inherent in the mobile environment.
- Explain the demands placed on the mobile operating system by the user.
- Discuss the challenges of implementation across mobile platforms.
- Discuss sources of security threats and their management.

CE-SRM-6 Support for concurrent processing
Minimum core coverage time: 3 hours

Core Learning Outcomes
- Explain and give examples of basic concepts in concurrent processing such as multiprocessor, multicore, SIMD, MIMD, shared memory, and distributed memory.
- Explain what is needed to support scheduling of multiple threads.
- Describe how simultaneous multithreaded (SMT) execution works.

CE-SRM Supplementary Knowledge Units

CE-SRM-7 System performance evaluation
Supplementary

Elective Learning Outcomes
- Describe why performance is important in significant applications (e.g., mission critical systems).
- Explain why metrics such as response time, throughput, latency, availability, reliability, and power consumption are significant in performance evaluation.
- Explain what must be measured to make use of important performance evaluation metrics.
- Describe the strengths of commonly used benchmark suites and their limitations.
- Demonstrate understanding of commonly used evaluation models (e.g., deterministic, stochastic, simulation) and circumstances in which it is appropriate to use them.
- Explain how profiling and tracing data is collected and used in evaluating system performance.

CE-SRM-8 Support for virtualization
Supplementary

Elective Learning Outcomes
- Define the role of the hypervisor or virtual machine monitor.
- Describe what the role of the host machine is and its relationship to the guest machines.
- Describe what the host operating system is and how it is related to guest operating systems.
- Explain what a native hypervisor is and how it differs from a hosted hypervisor.
- Give examples of isolation and security issues arising in virtualized environments.

CE-SWD Software Design

[45 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

1. Programming paradigms and constructs
2. Data structures and use of standard library functions for manipulating them
3. Object oriented design and the use of modeling languages
4. Testing and software quality concepts
5. Tradeoffs among different software design methods

CE-SWD Core Knowledge Units

CE-SWD-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Explain why early software was written in machine language and assembly language.
- Name some early programming languages and list some of their key features.
- Give examples of milestones in interactive user interfaces.
- Describe the magnitude of changes in software development environments over time.
- Explain why high level languages are important to improved productivity.
- List and define the steps of a software life cycle.

CE-SWD-2 Relevant tools, standards, and/or engineering constraints

Minimum core coverage time: 3 hours

Core Learning Outcomes:

- Identify the roles of software development tools, such as compilers, assemblers, linkers, and debuggers.
- Identify the expected functionality of a typical modern integrated development environment (IDE).
- Use an IDE to develop a simple application.
- Effectively use a debugger to trace code execution and identify defects in code.

CE-SWD-3 Programming constructs and paradigms

Minimum core coverage time: 12 hours

Core Learning Outcomes:

- Explain the execution of a simple program.
- Write simple and secure programs that accomplish the intended task.
- Write simple functions and explain the roles of parameters and arguments.
- Design, implement, test, and debug a program that uses fundamental programming constructs in nontrivial ways.
- Choose appropriate iteration and conditional constructs to accomplish a given programming task.
- Contrast imperative (i.e., procedural), declarative (i.e., functional), and structured (i.e., object-oriented) software design paradigms.
- Define and explain the elements of good programming (including the need to avoid opportunities for security breaches).

CE-SWD-4 Problem-solving strategies

Minimum core coverage time: 5 hours

Core Learning Outcomes:

- Identify a practical example of a problem in which different problem-solving strategies would be useful.
- Use a divide-and-conquer strategy to solve a problem.
- Use a greedy approach to solve an appropriate problem and determine if the approach used produces an optimal result.
- Explain the role of heuristics in problem-solving strategies.
- Discuss tradeoffs among different problem-solving strategies.
- Given a problem, determine an appropriate problem-solving strategy to use in devising a solution.

Elective Learning Outcomes:

- Use dynamic programming to solve an appropriate problem.

CE-SWD-5 Data structures
Minimum core coverage time: 5 hours

Core Learning Outcomes:
- Demonstrate knowledge of fundamental data structures, their uses, and tradeoffs among them.
- Demonstrate the use of high quality program libraries for building and searching data structures.
- Explain what a hash table is, why it is useful, and the role that collision avoidance and resolution play.
- Explain what a binary search tree is and why maintaining balance has impact on algorithm performance.
- Solve problems using library functions for standard data structures (linked lists, sorted arrays, trees, and hash tables) including insertion, deletion, searching and sorting (rather than implementing the algorithm from scratch).

CE-SWD-6 Recursion
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Describe the concept of recursion and give examples of its use.
- Explain the relationship between iteration and recursion.
- Identify the base case and the general case of a recursively defined problem.
- Itemize possible problems that can occur at run-time because of employing recursion in programs.

CE-SWD-7 Object-oriented design
Minimum core coverage time: 4 hours

Core Learning Outcomes:
- Decompose a problem domain into classes of objects having related state (data members) and behavior (methods).
- Contrast and contrast method overloading and overriding and illustrate with examples.
- State the benefits and disadvantages of compile-time vs. runtime method binding.
- Employ a modeling language (such as UML) to illustrate a simple class hierarchy with subclass structure that allows re-use of code for different subclasses.
- Explain mechanisms for disambiguation of function invocation when method names are overridden or overloaded.
- Explain the concepts of information hiding, coupling and cohesion, and data abstraction as they relate to object-oriented design.

CE-SWD-8 Software testing, verification, and validation
Minimum core coverage time: 5 hours

Core Learning Outcomes:
- Explain the differences between testing, verification, and validation.
- Demonstrate an understanding of unit testing strategies (e.g., white box, black box, and grey box) and tradeoffs.
- Demonstrate an understanding of verification and validation strategies.
- Construct a test dataset for use in unit testing of a module, exercise that dataset, and produce a test report.
- Explain the difference between unit testing and integration testing.
- Explain commonly used metrics for software quality.
- Describe at least one tool for automated testing and/or test pattern generation.

CE-SWD-9 Data modeling
Minimum core coverage time: 2 hours

Core Learning Outcomes:
- Explain and provide examples of data models and their use.
- Employ standard modeling notation (such as UML) to express and document an appropriate data model for a computer engineering problem.

CE-SWD-10 Database systems
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Explain how use of database systems evolved from programming with simple collections of data files.
- Describe the major components of a modern database system.
- Describe the functionality provided by languages such as SQL.
- Give examples of interactions with database systems that are relevant to computer engineering.

CE-SWD-11 Event-driven and concurrent programming
Minimum core coverage time: 2 hours

Core Learning Outcomes:
- Explain the problems associated with mobile systems and location aware systems, including the security issues.
- Explain the difficulties associated with the programming effective programming of multi-core processors.
- Demonstrate an ability to effectively program a multi-core processor.

CE-SWD Supplementary Knowledge Units

CE-SWD-12 Using application programming interfaces
Supplementary

Elective Learning Outcomes:
- Design and implement reusable functions.
- State the purpose of deprecation.
- Define backward and forward compatibility problems and some solutions to issues associated with these problems.
- Write code against common application program interfaces (APIs) for system services, data structures, or network communications.

CE-SWD-13 Data mining
Supplementary

Elective Learning Outcomes:
- Explain the role of data mining in computer engineering applications.
- Provide an illustration of the role of machine learning in computer engineering.

CE-SWD-14 Data visualization
Supplementary

Elective Learning Outcomes:
- Construct visualizations of data that improve comprehensive of the data, communication of the relevant information, and aid in decision making.

Appendix B

Computer Engineering Sample Curricula

This appendix to the *Computing Curricula - Computer Engineering (CE2016)* report contains several sample curricula illustrating possible implementations of degree programs each satisfying the required specifications of the body of knowledge detailed in this report. This appendix contains both three- and four-year programs. Although four-year programs are most common, these examples illustrate how undergraduate programs of different flavors and of different characteristics could implement these recommendations to suit different institutional requirements and resource constraints. Hence, they serve a wide variety of educational goals and student needs. None of these examples is prescriptive.

The following table summarizes the sample curricula in this appendix. This table serves as a guide to identifying sample curricula that are most relevant to specific institutional needs and priorities.

Curriculum	Administrative Entity
A	Electrical & Computer Engineering Department
B	Computer Science Department
C	Joint – Computer Science and Electrical Engineering Departments
D	People's Republic of China
E	Bologna-3

B.1 Format and Conventions

All sample curricula in this appendix use a common format with five logical components:

1. A set of educational objectives for the program of study and an explanation of any assumed institutional, college, department, or resource constraints
2. A summary of degree requirements, in tabular form, to indicate the curricular content in its entirety
3. A sample 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

B.1.1 Course Hour Conventions

To clarify the identification of courses, levels, and implementations, course numbers reflect the identity of the curriculum in which it appears and the level at which it appears in the program. Thus, a course numbered MTH_x100 is a course in curriculum X commonly taught in the first year (at the freshman level). Likewise, PHY_x200 is a course commonly taught in the second year (at the sophomore level); ECE_x300 is a course commonly taught in the third year (at the junior level); and course ECE_x400 is commonly taught in the fourth year (at the senior level).

To provide ease of comparison, all curricula implementations appear as a set of courses designed for a US system in which a semester provides 14 weeks of lecture and laboratory. Typically, there is the equivalent of one week for examinations, vacations, and reading periods. For simplicity, we specify lecture and lab times in “hours,” where one “hour” of lecture or lab is typically 50 minutes in duration.

We assign semester credit hours to each course, based on 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 = One, 1-hour lecture per week
- Laboratory hours: formal experimentation in a laboratory setting
  - 1 credit hour = One, 3-hour laboratory session per week

The following are examples of ways to calculate credits for lectures and laboratories where the word hour is a 50-minute time segment.

- *3-credit lecture course*:
  3 lecture hours per week for 14 weeks = 42 lecture hours (plus one week for examinations)
- *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 (plus one week for examinations)
  One 3-hour lab per week for 14 weeks = 42 lab hours
- *3-credit project design course*:
  1 classroom meeting per week for 14 weeks = 14 lecture hours (plus one week for examinations)
  2 credits of laboratory = 6 hours of laboratory per week for 14 weeks = 84 lab hours

B.1.2 Mapping of the computer engineering BoK to a sample curriculum

Each sample curriculum contains a table that maps the computer engineering BoK to the sample curriculum. The table rows contain course numbers with BoK knowledge areas (KA) as column headers. If an entry in a row is non-empty, then it contains the numbered knowledge units (KUs) from the knowledge area covered in that course. For example, the entry 1-3,5 under the DIG knowledge area says that this course covers knowledge units 1,2,3 and 5 from CE-DIG. Note that:

- A course may have KUs from one KA, or it may have KUs from multiple KAs.
- The same KU may appear in multiple courses. For example, a two-course sequence in digital design may both contain the DIG-1,2 KUs as both courses may cover history and relevant tools/standards/constraints, but from different perspectives.

The second table row contains the minimum core BoK hours for each knowledge area as a reference. The bottom two rows, labeled *Core BoK Units* and *Supplementary BoK Units,* list the knowledge units from each knowledge area covered by this sample curriculum. Since all the sample curricula have complete BoK core coverage, the row labeled *Core BoK Units* contains all core knowledge units from the knowledge areas. The row labeled *Supplementary BoK Units* may contain non-empty entries, which list the supplementary BoK knowledge units covered in those knowledge areas. The sample curricula do not cover all the supplementary BoK knowledge units and the coverage shown does not convey a priority or recommended coverage.

B.1.3 Course descriptions

The provided course descriptions are what might typically appear in a course catalog. Because of their length, the topics listed in these short descriptions are not exhaustive. A list of the BoK knowledge areas and knowledge units covered by these courses augment these descriptions. For courses that include laboratory hours, these descriptions do not include details on the laboratory experience. Section 4.4 describes expectations for the overall laboratory experience, including teamwork, data collection and analysis, and other skills.

B.2 Preparation to Enter the Profession

A major goal of the sample curricula in this appendix is the preparation of graduates for entry into the computer engineering profession. There are many ways of building an undergraduate curriculum whose graduates are well-educated computer engineers. To emphasize this point, the programs of study outlined in this section are quite distinct. These programs differ in their emphasis and in the institutional constraints but do not represent all likely program structures. For example, many programs include a first-year course to introduce students to the discipline, provide hands-on engineering experiences, and engage the students. The absence of such a course in any of the example curricula is not a judgment on the value of such courses.

The CE2016 steering committee designed these curricula to ensure appropriate coverage of the core learning outcomes of the computer engineering BoK as defined in this report. However, as also discussed in the main report, there are many other elements in the creation of 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 their accrediting agency for complete accreditation criteria.

In addition, each individual computer engineering program may have educational goals 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 goals 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 common in the US. The remaining sample curricula (Curricula D and E) illustrate typical programs in computer engineering that one might find in China, the United Kingdom, Europe, Asia, and other parts of the world. Curriculum E is consistent with the Bologna Declaration, with the three-year program leading to a typical bachelor of engineering degree.

The common requirements are 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 broad-based course of study and provides selection from among many professional electives. In all cases, the culminating 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 goal of each program is to prepare students for a professional career in computer engineering by establishing a foundation for lifelong learning and development. Each program also provides a platform for further work leading to graduate studies in computer engineering, as well as careers in fields such as 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 culminating design course(s), which complements the analytical part of the curriculum.

Graduates of each program should be well prepared for professional employment or advanced studies, understanding 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 each of these computer engineering curricula includes the broad education necessary to understand the effect of engineering solutions in a global and societal context.

B.4 Curriculum A: Administered by Electrical and Computer Engineering

Computer Engineering Program Administered by an Electrical and Computer Engineering Department

B.4.1 Program Goals and Features

This program leads to a bachelor's degree in computer engineering, as might be offered by a traditional electrical and computer engineering (ECE) department. A computer science department sometimes offers foundation courses in computer science such as programming; the ECE department teaches the remaining courses. As is typical of most programs in engineering, this program has a smaller general education component than Curriculum B (described next), with more hours devoted to computer engineering areas. This program usually has multi-course sequences in circuits/electronics, digital design, embedded systems, and computer programming. Breadth coverage occurs through courses in computer architecture, operating systems, networks, and computer security. This program is oriented more towards hardware and system design, but contains more than sufficient software coverage to enable graduates to be effective computer engineers.

B.4.2 Summary of Requirements

This program of study contains five required computer science courses (17 credits) and 14 required electrical and computer engineering (ECE) courses (42 credits). The two computer engineering electives (6 credits), chosen from courses in either the computer science or electrical and computer engineering departments, provide flexibility in the program. Lab hours are present in ten courses, giving students significant hands-on experiences with modern tools and design techniques. The capstone experience occurs over two courses in the senior year, allowing for a substantial and complete design experience. Required oral technical writing courses address oral and written communication skills, reinforced throughout the laboratory and the capstone project courses. Credit hours are distributed as follows.

Credit-hours	*Areas*
21	Mathematics
10	Natural science (Physics, Chemistry)
24	English composition, humanities, and social sciences
38	Required electrical and computer engineering
17	Required computer science
6	Technical electives (from ECE or CSC)
4	Design project
0	Free electives
120	*TOTAL Credit Hours for Computer Engineering Program*

B.4.3 Four-Year Model for Curriculum A

CE: can be offered in computer engineering department
ECE: offered in the electrical and computer engineering department
CSC: offered in the computer science department
*CE technical electives: approved elective course in either department

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 1			Semester 2		
MTH 101	Calculus I	3	MTH 102	Calculus II	3
CHM 101	Chemistry I & Lab	4	PHY 101	Physics I	3
CSC_A101	Introduction to Computer Programming	4	CSC_A102	Intermediate Computer Programming	4
	English Composition I	3	ECE_A101	Introduction to ECE	2
	Humanities Elective	3		English Composition II	3
	Total Credit Hours	17		*Total Credit Hours*	15
Semester 3			Semester 4		
MTH 201	Calculus III	3	MTH 203	Differential Equations	3
PHY 201	Physics II	3	MTH_A204	Discrete Structures	3
ECE_A201	Digital Devices & Lab	4	ECE_A202	Microprocessors & Lab	4
CSC_A201	Data Structures	3	ECE_A203	Circuits/Electronics I	3
MTH 202	Linear Algebra	3	MTH 205	Probability & Statistics	3
	Total Credit Hours	16		*Total Credit Hours*	16
Semester 5			Semester 6		
CSC_A301	Algorithms	3	CSC_A302	Client/Server Programming	3
ECE_A301	Circuits/Electronics II & Lab	4	ECE_A303	Signals & Systems	3
ECE_A302	Digital System Design & Lab	3	ECE_A304	Data Communication	3
	Humanities Elective	3	ECE_A305	Computer Architecture	3
				Social Science Elective	3
	Total Credit Hours	13		*Total Credit Hours*	15
Semester 7			Semester 8		
ECE_A401	CE Design I	2	ECE_A402	CE Design II	2
ECE_A403	Embedded Systems & Lab	3	ECE_A404	Computer Security	3
ENG 401	Writing for Engineers	3	ECE_A405	Operating Systems	3
	CE Technical Elective*	3		CE Technical Elective*	3
	Fine Arts Elective	3		Social Science Elective	3
		14			14

B.4.4 Mapping of Computer Engineering BoK to Curriculum A

Refer to section B.1.2 for an explanation of this table.

BoK Area / Course	CAE	CAL	CAO	DIG	ESY	NWK	PPP	SEC	SGP	SPE	SRM	SWD	ACF	DSC	LAL	PRS
Minimum Core BoK Hours	**50**	**30**	**60**	**50**	**40**	**20**	**20**	**20**	**30**	**35**	**20**	**45**	**30**	**30**	**20**	**30**
CSC_A101												1-4				
CSC_A102												4-8				
CSC_A201												5-9				
CSC_A301		1-8														
CSC_A302										1-12		10,12				
ECE_A101							1-3,5									
ECE_A201				1-9												
ECE_A202			3, 5	3	1-8											
ECE_A203	1-4															
ECE_A301	5-10															
ECE_A302				1, 2, 6-11												
ECE_A303									1-7							
ECE_A304						1-11										
ECE_A305			1-11													
ECE_A401							1-6, 11			7-10						
ECE_A402							7-11			10-12						
ECE_A403					9-13					10	4,6	11				
ECE_A404								1-11								
ECE_A405		9									1-8					
MTH 202															1-10	
MTH 203													1-7			
MTH_A204														1-9		
MTH 205																1-10
Core BoK Units Covered	**1-10**	**1-8**	**1-11**	**1-11**	**1-13**	**1-11**	**1-11**	**1-11**	**1-7**	**1-12**	**1-6**	**1-10**	**1-7**	**1-9**	**1-10**	**1-10**
Supplementary BoK Units		**9**									**7-8**	**11,12**				

B.4.5 Curriculum A – Course Summaries

CSC_A101: Introduction to Computer Programming

Introductory problem solving and computer programming using object-oriented techniques; theoretical and practical aspects of programming and problem solving

Prerequisite: College Algebra or equivalent
Credit Hours: 4; Lecture Hours: 42; Lab Hours: 42
BoK Coverage: CE-SWD 1-4

CSC_A102: Intermediate Computer Programming

Object-oriented problem solving, design, and programming; introduction to data structures, algorithm design and complexity

Prerequisite: CSC_A101
Credit Hours: 4; Lecture Hours: 42; Lab Hours: 42
BoK Coverage: CE-SWD 4-8

CSC_A201: Data Structures & Analysis of Algorithms

Non-linear data structures and their associated algorithms; trees, graphs, hash tables, relational data model, file organization; advanced software design and development

Prerequisite: CSC_A102 and College Algebra or equivalent
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-SWD 5-9

CSC_A301: Introduction to Algorithms

Study of complexity of algorithms and algorithm design; tools for analyzing efficiency; design of algorithms, including recurrence, divide-and-conquer, dynamic programming and greedy algorithms

Prerequisite: CSC_A201, MTH 201 and MTH_A204
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-CAL 1-8

CSC_A302: Distributed Client/Server Programming

Design of software systems for use in distributed environments; client/Server models, multithreaded programming, server-side web programming, graphical user interfaces, group projects involving client/server systems

Prerequisite: CSC_A201
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 42
BoK Coverage: CE-SWD 10, CE-SPE 1-12; CE-SWD 12

ECE_A101: Introduction to ECE

What it means to be an engineer, engineering ethics, engineering modeling, the design process, areas of ECE, communication skills

Prerequisite: Credit or registration in MTH 101
Credit Hours: 2; Lecture Hours: 15; Lab Hours: 42
BoK Coverage: CE-PPP 1-3, 5

ECE_A201: Digital Devices

Binary codes, Boolean algebra, combinational logic design, flip-flops, counters, synchronous sequential logic, programmable logic devices, MSI logic devices, adder circuits

Prerequisite: Credit or registration in CSC_A101
Credit Hours: 4; Lecture Hours: 42; Lab Hours: 42
BoK Coverage: CE-DIG 1-9

ECE_A202: Microprocessors

Architecture of microprocessor-based systems; study of microprocessor operation, assembly language, arithmetic operations, and interfacing

Prerequisite: ECE_A201, CSC_A201
Credit Hours: 4; Lecture Hours: 42; Lab Hours: 42
BoK Coverage: CE-ESY 1-8, CE-CAO 3, 5, CE-DIG 3

ECE$_A$203: Introduction to Electronic Circuits

Fundamentals of electric circuits and network analysis; transient analysis and frequency response of networks; introduction to operational amplifiers; AC power

Prerequisite: PHY 201, MTH 202, Credit or registration in MTH 203
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-CAE 1-4

ECE$_A$301: Intermediate Electronic Circuits

Operation circuit models and application of diodes and field-effect and bipolar junction transistors; electronic instrumentation; foundations of electrical communications systems

Prerequisite: ECE$_A$203
Credit Hours: 4; Lecture Hours: 42; Lab Hours: 42
BoK Coverage: CE-CAE 5-10

ECE$_A$302: Digital System Design

Modern digital design techniques using logic synthesis, hardware description languages; field programmable gate arrays, and modular building blocks

Prerequisite: ECE$_A$202
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 42
BoK Coverage: CE-DIG 1, 2, 6-11

ECE$_A$303: Signals and Systems

Modeling of analog and discrete-time signals and systems, time domain analysis; Fourier series, continuous and discrete-time Fourier transforms and applications, sampling, z-transform, state variables

Prerequisite: ECE$_A$301
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-SGP 1-7

ECE$_A$304: Data Communication Networks

The concepts and practices of data communications and networking to provide the student with an understanding of the hardware and software used for data communications

Prerequisite: ECE$_A$202
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-NWK 1-11, CE-SET 3-5

ECE$_A$305: Computer Architecture

Detailed design and implementation of a stored-program digital computer system; designs for the CPU, I/O subsystems, and memory organizations; ALU design and computer arithmetic

Prerequisite: ECE$_A$202
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-CAO 1-11

ECE$_A$401: CE Design I

Lectures on teaming, project management, engineering standards, economics, and ethical and professional issues; student selects faculty mentor, performs project design, and presents orally

Prerequisite: ECE$_A$302, CSC$_A$302
Credit Hours: 2; Lecture Hours: 14; Lab Hours: 45
BoK Coverage: CE-SPE 7-10, CE-PPP 1-6, 11

ECE$_A$402: CE Design II

Lectures on teaming, project management, engineering standards, economics, and ethical and professional issues; student selects faculty mentor, performs project design, and presents orally

Prerequisite: ECE$_A$401
Credit Hours: 3; Lecture Hours: 14; Lab Hours: 42
BoK Coverage: CE-SPE 10-12, CE-PPP 7-11

ECE_A403: Embedded Systems

Advanced challenges in embedded systems design using contemporary practice; interrupt-driven, reactive, real-time, object-oriented, and distributed client/server embedded systems

Prerequisite: ECE_A302, ECE_A301
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 42
BoK Coverage: CE-ESY 9-13, CE-SPE 10, CE-SRM 4, 6; Supplementary CE-SWD 11

ECE_A404: Information Security

Basic and advanced concepts in cryptography and network security: symmetric and asymmetric cryptography, key management, wired and wireless network security protocols, network systems security

Prerequisite: ECE_A304
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-SEC 1-11

ECE_A405: Operating Systems

Historical development of operating systems to control complex computing systems; process management, communication, scheduling techniques; file systems concepts and operation; data communication, distributed process management

Prerequisite: ECE_A202
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-SET 3-5, CE-SRM 1-6; Supplementary CE-CAL 9, CE-SRM 7-8

MTH_A204: Discrete Structures

Concepts of algorithms, induction, recursion, proofs, logic, set theory, combinatorics, graph theory fundamental to study of computer science

Prerequisite: CSC_A101 and College Algebra or equivalent
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-DSC 1-7

B.5 Curriculum B: Administered by Computer Science

Computer Engineering Program Administered by a Computer Science Department

B.5.1 Program Goals and Features

A computer science department sponsors this bachelor of science program in computer engineering. Programs of this kind often develop through evolution from a computer science program, and therefore this model might be of interest to schools that have a computer science department and either a general engineering program or a physics department that teaches introductory engineering courses. As is typical of many computer science programs, this model has a significantly larger general education component than the other curricula presented in this appendix, and therefore fewer hours devoted to computer engineering. For this reason, the CE2016 steering committee designed several courses specifically to provide coverage of the core learning outcomes of the CE BoK. Specifically, the committee designed courses that cover traditional electrical engineering in the CE core to cover the core material without providing significant breadth or depth beyond the core learning outcomes. In contrast, several computer science courses do go somewhat beyond the core material. There is, however, sufficient coverage of electrical engineering, hardware, and systems to enable graduates to be effective computer engineers.

B.5.2 Summary of Requirements

This program of study builds around a set of eleven required courses in computer science (including a culminating design project) and three in electrical engineering. The program achieves flexibility through a judicious choice of three technical electives and a culminating project. Laboratory experience occurs in the first two introductory computer science courses, in the circuits and electronics course, and in the digital logic course. The total number of hours devoted to laboratory experience is less than in the other curriculum models presented in this appendix. In addition, since there are fewer courses that incorporate engineering design, the culminating design project experience extends two full semesters to ensure that all graduates have significant design experience, as well as experience with teamwork and modern engineering tools. The CE2016 steering committee assumes that oral and written communication skills occur in both the general education and in the computer engineering segments of this curriculum.

This curriculum utilizes a relatively traditional course structure and content, with credit hours distributed as follows.

Credit-hours	*Areas*
20	Mathematics and statistics
11	Natural science (physics, chemistry)
33	Humanities, social sciences, composition, and literature
25	Required computer science (excluding design project)
11	Required electrical engineering
9	Technical electives (from computer science or engineering)
5	Culminating design project (from computer science)
6	Free electives
120	*TOTAL Credit Hours for Computer Engineering Program*

B.5.3 Four-Year Model for Curriculum B

ELE: offered in an engineering or physics department
CSC: offered in the computer science department
*CE Technical Electives: offered in an engineering or computer science department

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 1			**Semester 2**		
MTH 101	Calculus I	4	MTH 102	Calculus II	4
CHM 101	Chemistry I	4	PHY 101	Physics I	4
CSC_B101	Computer Science I	3	CSC_B102	Computer Science II	3
	English Composition	3		Humanities / Social Science	3
	Total Credit Hours	*14*		*Total Credit Hours*	*14*
Semester 3			**Semester 4**		
MTH 201	Differential Equations	3	MTH 203	Linear Algebra	3
PHY 201	Physics II	3	CSC_B201	Algorithm Design	3
MTH_B202	Discrete Structures	3	ELE_B202	Circuits and Electronics	4
ELE_B201	Introduction to Digital Design	4	CSC_B202	Computer Organization	3
	Humanities / Social Science	3		Humanities / Social Science	3
	Total Credit Hours	*16*		*Total Credit Hours*	*16*
Semester 5			**Semester 6**		
MTH 301	Probability and Statistics	3	CSC_B302	Embedded Systems	3
CSC_B301	Computer Architecture	3	CSC_B303	Computer Networks & Information Security	3
ELE_B301	Systems and Signal Processing	3	CSC_B304	Operating Systems	3
	Technical Writing	3		Humanities / Social Science	3
	Humanities / Social Science	3		Humanities / Social Science	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 7			**Semester 8**		
CSC_B401	Senior Project I	2	CSC_B402	Senior Project II	3
CSC_B403	Ethics and Professionalism	1		CE Technical Elective	3
	CE Technical Elective	3		CE Technical Elective	3
	Humanities / Social Science	3		Humanities / Social Science	3
	Humanities / Social Science	3		Free Elective	3
	Free Elective	3			
	Total Credit Hours	*15*		*Total Credit Hours*	*15*

B.5.4 Mapping of Computer Engineering BoK to Curriculum B

Refer to section B.1.2 in the front of this appendix for an explanation of this table.

BoK Area / Course	CAE	CAL	CAO	DIG	ESY	NWK	PPP	SEC	SGP	SPE	SRM	SWD	ACF	DSC	LAL	PRS
Minimum Core BoK Hours	**50**	**30**	**60**	**50**	**40**	**20**	**20**	**20**	**30**	**35**	**20**	**45**	**30**	**30**	**20**	**30**
CSC_B101												1-5				
CSC_B102												1, 6-10, 11-12				
CSC_B201		1-8, 10														
CSC_B202			1-4, 6-8													
CSC_B301			6-11	8, 10												
CSC_B302					1-12											
CSC_B303						1-8		1-11								
CSC_B304											1-6, 7-8					
CSC_B401							2-4, 6-9			1-12						
CSC_B402							2-4, 6-9			1-12						
CSC_B403							1, 5, 6-10									
ELE_B201			5	1-7, 9, 11												
ELE_B202	1-10															
ELE_B301									1-8							
MTH 101													1-3			
MTH 102													3-4			
MTH 201													5-7			
MTH_B202														1-9		
MTH 203															1-10	
MTH 301																1-9
Core BoK Units Covered	**1-10**	**1-8**	**1-11**	**1-10**	**1-12**	**1-8**	**1-9**	**1-11**	**1-8**	**1-12**	**1-6**	**1-10**	**1-7**	**1-9**	**1-10**	**1-9**
Supplementary BoK Units		**10**		**11**							**7-8**	**11-12**				

B.5.5 Curriculum B – Course Summaries

CSC_B101: Computer Science I

Introduction to computing; algorithmic thinking, problem solving in the context of a modern programming language and its associated development environment

Prerequisites: Pre-calculus or equivalent
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 42
BoK Coverage: CE-SWD 1-5

CSC_B102: Computer Science II

Second course in programming languages and systems; object-oriented design, data structures, recursion, data modeling, fundamental concepts in software engineering

Prerequisites: CSC_B101
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 42
BoK Coverage: CE-SWD 1, 6-10; Supplementary CE-SWD 11-12

CSC_B201: Algorithm Design

Analysis and design of algorithms, algorithm design strategies, searching and sorting algorithms, parallel algorithms, tradeoffs in algorithmic performance, algorithmic complexity

Prerequisites: CSC_B102
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-CAL 1-8; Supplementary CE-CAL 10

CSC_B202: Computer Organization

Introductory course in computer organization and architecture; processor organization, instruction set architecture, memory system organization, performance, and interfacing fundamentals

Prerequisites: CSC_B101
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-CAO 1-4, 6-8

CSC_B301: Computer Architecture

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

Prerequisites: CSC_B202
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-CAO 6-11; CE-DIG 8, 10

CSC_B302: Embedded Systems

Characteristics of embedded systems, techniques for embedded applications, parallel input and output, synchronous and asynchronous serial communication, interrupt handling, applications involving data acquisition, control, sensors, and actuators, implementation strategies for complex embedded systems

Prerequisites: CSC_B202, CSC_B301
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 42
BoK Coverage: CE-ESY 1-12

CSC_B303: Computer Networks and Information Security

Introduction to the design and performance analysis of local computer networks; architectures, protocols, standards, and technologies of computer networks; principles of information security, authentication, sources of vulnerability, malware, defenses against attack, network security

Prerequisites: CSC_B202, MTH_B202; Co-Requisite CSC_B304
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-NWK 1-8; CE-SEC 1-11

CSC_B304: Operating System

Basic operating systems and their components; concurrency, scheduling and dispatch, memory and device management, file systems and performance evaluation, real-time operating systems, operating systems for mobile

devices

Prerequisites: CSC_B201, CSC_B202
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 42
BoK Coverage: CE-SRM 1-6; Supplementary CE-SRM 7-8

CSC_B401: Senior Project I

Individually designed projects oriented toward providing experience in project management, risk management, specification of system requirements and architecture, system design, integration, testing, and deployment; solution of open-ended problems; emerging technologies discussed in the context of these projects

Prerequisites: CSE_B302, CSE_B312, CSE_B332
Credit Hours: 2; Lecture Hours: 14; Lab Hours: 42
BoK Coverage: CE-PPP 2-4, 6-9, CE-SPE 1-12

CSC_B402 Senior Project II

Continuation of Senior Project I focused on implementation of a project design

Prerequisites: CSC_B401
Credit Hours: 3; Lecture Hours: 14; Lab Hours: 84
BoK Coverage: CE-PPP 2-4, 6-9, CE-SPE 1-12

CSC_B403: Ethics and Professionalism

Critical examination of ethical problems associated with computer science and engineering; legal and quasi-legal (i.e., policy and regulative) issues are also considered; 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: 0; Lab Hours: 42
BoK Coverage: CE-PPP 1, 5, 6-10

ELE_B201: Introduction to Digital Design

Number systems and representation of information; computer arithmetic; analysis and synthesis of combinational and sequential logic circuits; use of a modern hardware description language; organization and structure of computing systems

Prerequisites: CSC_B101
Credit Hours: 4; Lecture Hours: 42; Lab Hours: 42
BoK Coverage: CE-DIG 1-7, 9, CE-CAO 5; Supplementary CE-DIG 11

ELE_B202: Circuits and Electronics

Fundamentals of electric circuits and network analysis; transient analysis, frequency response, Laplace transforms, Fourier series, introduction to electronic materials and devices, diodes, bipolar transistors and logic families, MOS technology

Prerequisites: MTH 201, PHY 102
Credit Hours: 4; Lecture Hours: 42; Lab Hours: 42
BoK Coverage: CE-CAE 1-10

ELE_B301: Systems and Signal Processing

Sinusoidal and transient analysis, convolution, transform analysis, frequency response, digital processing of signals, difference equations, sampling and aliasing, discrete time transforms and digital filter design

Prerequisites: ELE_B202
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-SGP 1-8

MTH_B202: Discrete Structures

Sets, functions, and relations, Boolean algebra, first order logic, proof techniques, counting arguments, iteration and recursion, graphs, and trees

Prerequisites: Pre-calculus or equivalent
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-DSC 1-9

B.6 Curriculum C: Administered jointly by CS and EE

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 (CSC) department and an electrical and computer engineering (ECE) department jointly administer this program leading to a bachelor's degree in computer engineering (CE). This curriculum brings a core competency and unique value of integrated knowledge in both computer software and hardware, providing a balance among computer systems, hardware, and software as well as theory and applications. Studies in computer engineering integrate fields from both computer science and electrical engineering. Computer engineering students receive a flexible curriculum that has a balanced common core of required courses. Specialization occurs via technical elective courses chosen from either department, tailoring a curriculum to a student's interest and employers' needs in a dynamic job market. 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 to pursue advanced studies in computer engineering or they can choose from many different careers related to computers and their applications in high technology environments.

B.6.2 Summary of Requirements

The general education component of this curriculum consists of 21 credits of mathematics, 16 credits of natural science, and 18 credits of humanities and social science courses. The required core of computer engineering courses consists of traditional CSC courses (e.g., programming fundamentals), traditionally ECE courses (e.g., circuits & electronics), and CE courses that can be taught by computer engineering faculty members in either department (e.g., computer architecture & organization). Specialization occurs via technical electives chosen in various knowledge areas from either department. Finally, the culminating design experience is a two-course sequence in the senior year (CE Design I and CE Design II). Credit hours are distributed as follows.

Credit-hours	*Areas*
21	Mathematics
16	Natural science (physics, chemistry)
18	Humanities, social sciences, composition, literature
12	Required computer engineering (CE)
19	Required electrical and computer engineering (ECE)
16	Required computer science (CSC)
12	Technical electives
6	Design project
0	Free electives
120	*TOTAL Credit Hours for Computer Engineering Program*

B.6.3 Four-Year Model for Curriculum C

CE: can be offered in computer engineering department
ECE: offered in the electrical and computer engineering department
CSC: offered in the computer science department
*CE Technical Electives: elective course offered in either department

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 1			**Semester 2**		
MTH 101	Calculus I	4	MTH 102	Calculus II	4
CHM 101	Chemistry I + Lab	4	PHY 101	Physics I + Lab	4
	Humanities / Social Science	3	CSC_C101	Programming Fundamentals I	3
	Humanities / Social Science	3	CHM 102	Chemistry II or Biological Science	4
	Total Credit Hours	*14*		*Total Credit Hours*	*15*
Semester 3			**Semester 4**		
MTH 201	Calculus III	4	MTH 202	Differential Equations	3
PHY 201	Physics II + Lab	4	ECE_C202	Circuits & Electronics	4
CSC_C201	Programming Fundamentals II	3	CE_C201	Intro to Computer Architecture & Organization	3
ECE_C201	Digital & Logic Design + Lab	4	CSC_C202	Intro to Discrete Structures	3
				Humanities / Social Science	3
	Total Credit Hours	*15*		*Total Credit Hours*	*16*
Semester 5			**Semester 6**		
MTH 301	Computational Linear Algebra	3	CE_C301	Introduction to Computer Systems Engineering	3
CSC_C301	Data Structures & Algorithms	4	MTH 302	Engineering Statistics	3
ECE_C301	Embedded & Microprocessor Systems + Lab	4	CE_C302	Computing Networks	3
ECE_C302	Digital System Design + Lab	4		CE Technical Elective*	3
				Humanities / Social Science	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 7			**Semester 8**		
CSC_C401	Operating Systems	3	CE_C401	Computer and Network Security	3
ECE_C401	Signals & Systems	3		CE Technical Elective*	3
CE_C402	CE Design I	3		CE Technical Elective*	3
	CE Technical Elective*	3	CE_C403	CE Design II	3
	Humanities / Social Science	3		Professional Communication & Ethics for Engineers	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*

B.6.4 Mapping of Computer Engineering BoK to Curriculum C

Refer to section B.1.2 for an explanation of this table.

BoK Area / Course	CAE	CAL	CAO	DIG	ESY	NWK	PPP	SEC	SGP	SPE	SRM	SWD	ACF	DSC	LAL	PRS
Minimum Core BoK Hours	**50**	**30**	**60**	**50**	**40**	**20**	**20**	**20**	**30**	**35**	**20**	**45**	**30**	**30**	**20**	**30**
CSC_C101												1-5				
CSC_C201		4										3-7, 12				
CSC_C202														1-9		
CSC_C301		1-8										4, 5				
CSC_C401											1-7					
ECE_C201				1-9												
ECE_C202	1-10															
ECE_C301			6-8		1-8											
ECE_C302				2, 6-10						8-9						
ECE_C401									1-8							
CE_C201		6	1-9													
CE_C301							2,6			1-12	3	8-10				
CE_C302		6				1-8		8								
CE_C401								1-11								
CE_C402							1-4									
CE_C403							2-10									
MTH 202													1-7			
MTH 301															1-10	
MTH 302																1-9
Core BoK Units covered	**1-10**	**1-8**	**1-11**	**1-10**	**1-8**	**1-8**	**1-10**	**1-11**	**1-8**	**1-12**	**1-6**	**1-10**	**1-7**	**1-9**	**1-10**	**1-9**
Supplementary BoK Units											**7**	**12**				

B.6.5 Curriculum C – Course Summaries

CSC_C101: Programming Fundamentals I

Introductory problem solving and computer programming concepts, including object-oriented programming, procedural and data abstraction, and program modularity

Prerequisite: none
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-SWD 1-5

CSC_C201: Programming Fundamentals II

Intermediate problem solving and computer programming concepts, including algorithmic strategies, recursion, and effective design and use of data structures and application programming interfaces (APIs)

Prerequisite: CSC_C101
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-CAL 4, CE-SWD 3-7; Supplementary CE-SWD 12

CSC_C202: Introduction to Discrete Structures

Concepts of discrete structures including Boolean algebra, first-order algebra, proof techniques, set theory, and graph theory

Prerequisite: CSC_C201, MTH 101
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-DSC 1-9

CSC_C301: Data Structures and Algorithms

Design of basic data structures (e.g., stacks, queues, heaps, link structures, trees, graphs) and their manipulation; design and analysis of classic algorithms for common tasks (e.g., sorting, searching, graph algorithms)

Prerequisite: CSC_C201
Credit Hours: 4; Lecture Hours: 56; Lab Hours: 0
BoK Coverage: CE-CAL 1-8, CE-SWD 4-5

CSC_C401: Operating Systems

Basic operating systems and their components: scheduling, resource management, process management, interrupt handling, concurrent processing, and system performance evaluation

Prerequisite: CSC_C301, CE_C201
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-SRM 1-6; Supplementary CE-SRM 7

ECE_C201: Digital & Logic Design + Lab

Elements of digital design (e.g., Boolean algebra, basic logic circuits), analysis and synthesis of combinational and sequential logic circuits, introduction into finite state machines, hardware description language (HDL), and programmable logic devices (e.g., FPGAs)

Prerequisite: CSC_C101
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-DIG 1-9

ECE_C202: Circuits & Electronics

Fundamentals of circuit analysis: basic concepts (e.g., voltage, current, Ohm's law); transient analysis and frequency response of networks; introduction to operational amplifiers, electronic materials and devices, bipolar transistors and logic families, and MOS technology

Prerequisite: MTH 201, PHY 201
Credit Hours: 4; Lecture Hours: 56; Lab Hours: 0
BoK Coverage: CE-CAE 1-10

ECE_C301: Embedded & Microprocessor Systems + lab

Microprocessor-based embedded systems, synchronous and asynchronous serial communication, interfacing, interrupt handling, data acquisition, real-time processing

Prerequisite: CE_C201, ECE_C201
Credit Hours: 4; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-CAO 6-8, CE-ESY 1-8

ECE_C302: Digital System Design + Lab

Advanced digital design: modular design of combinational and sequential logic building blocks, control-datapath; extensive use of hardware description language (HDL) and programmable logic devices (e.g., FPGAs); system architecture design and evaluation

Prerequisite: ECE_C201, CE_C201
Credit Hours: 4; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-DIG 2, CE-DIG 6-10, CE-SPE 8-9

ECE_C401: Signals & Systems

Continuous-time and discrete-time signal analysis including Fourier series and discrete-time Fourier transforms; sampling; finite and infinite impulse response (FIR and IIR) filter design; frequency response, and system function

Prerequisite: MTH 201, MTH 202, MTH 301
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-SGP 1-8

CE_C201: Introduction to Computer Architecture & Organization

Introduction to computer architecture and system organization including instruction set architecture, processor organization, bus structures, memory sub-systems, input/output interfacing and communication, pipelining, and performance measurement

Prerequisite: CSC_C201, ECE_C201
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-CAO 1-9, CE-CAL 6

CE_C301: Introduction to Computer Systems Engineering

Fundamental concepts in computer system engineering including project management; architectural design; concurrent hardware and software design; system integration, testing, and validation; and concepts such as maintainability, sustainability, manufacturability

Prerequisite: CE_C201, CSC_C301
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-SPE 1-12, CE-PPP 2, CE-PPP 6, CE-SRM 3, CE-SWD 8-10

CE_C302: Computer Network

Architectures, protocols, standards, and technologies of computer networks including local and wide area networks, wireless and mobile networks, network applications, and network management and security

Prerequisite: CE_C201, ECE_C301
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-CAL 6, CE-NWK 1-8, CE-SEC 8

CE_C401: Computer and Network Security

Concepts in computer and network security including data security and integrity, vulnerabilities and exploitation, social engineering, cryptography, authentication, network and web security, and trust computing

Prerequisite: CE_C201, CE_C302
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-NWK 8, CE-SEC 1-11

CE_C402: CE Design I

First of a two-semester computer engineering capstone project which cumulates in a product with emphasis on teaming, project management, concurrent hardware and software design; system integration, testing, and validation

Prerequisite: CE_C301, ECE_C301
Credit Hours: 3; Lecture Hours: 14; Lab Hours: 84
BoK Coverage: CE-PPP 1-4

CE_C403: CE Design II

Second of a two-semester computer engineering capstone project which cumulates in a product with emphasis on teaming, project management, concurrent hardware and software design; system integration, testing, and validation

Prerequisite: CE_C402
Credit Hours: 3; Lecture Hours: 14; Lab Hours: 84
BoK Coverage: CE-PPP 2-10

B.7 Curriculum D: Administered in China

Computer Engineering Program Administered by the Department of Computer Science and Technology in China

B.7.1 Program Goals and Features

This program leads to a bachelor's degree in computer science and technology (CST). A computer science and technology department generally combines computer engineering with computer science in China and administers the program for the students in the computer engineering track. The curriculum brings together computing theory, computer architecture, computer networks, and computer programming. It covers the core requirements of a CST degree, but emphasizes understanding of the hardware and software from a system perspective. The program provides a broad background in computer engineering concepts and practice that prepares students for graduate study or careers in a wide range of industries and organizations.

B.7.2 Summary of Requirements

This program of study contains 22 required computer science and technology courses. The program provides flexibility through ten hours of technical electives chosen from courses in either the electrical engineering or the computer science and technology departments. Four laboratory-oriented courses, plus laboratory hours in other courses, provide students significant hands-on experiences with modern tools and design techniques. The capstone project for graduation is in the last semester to ensure that all graduates have significant design experience. The program addresses oral and written communication skills throughout the laboratory and the capstone project courses. Credit hours are distributed as follows.

Credit-hours	*Areas*
25	Mathematics
10	Natural science (physics, chemistry)
2	Elective mathematics and natural Science
41	Humanities, social sciences, composition, literature, foreign language, physical education
48	Required computer science and engineering (CE)
12	Lab courses – computer science and engineering (CE)
10	Technical Electives – basic and advanced (CE)
15	Capstone project for graduation (CE)
0	Free electives
163	*TOTAL Credit Hours for Computer Engineering Program*

B.7.3 Four-Year Model for Curriculum D

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 1			Semester 2			Semester 3		
	Physical Education I	1		Physical Education II	1		Practice in English	2
	Humanities and social I	3		Humanities and social II	3	CST_D 301	Practice in Programming (Lab Course)	2
	English language I	2		English language II	2			
MTH 101	Calculus I	5	MTH 201	Calculus II	5			
MTH 102	Linear Algebra I	4	MTH 202	Linear Algebra II	2			
MTH 103	Discrete Mathematics I	3	CST_D 201	Fundamentals Object-oriented Programming	2			
CST_D 101	Fundamentals of Programming	3	MTH 203	Discrete Mathematics II	3			
CST_D 102	Introduction of Information Science	1	PHY 201	Physics I	4			
	Elective of Humanities	1						
	Total Credit Hours	23		*Total Credit Hours*	22		*Total Credit Hours*	4
Semester 4			Semester 5			Semester 6		
PHY 401	Physics II	4	CST_D 501	Digital Logic Circuits & Lab	4	CST_D 601	Practice in Assembly Language (Lab Course)	3
PHY 402	Lab of Physics I	1	PHY 501	Lab of Physics II	1	CST_D 602	Practice in Java (Lab Course)	2
CST_D 401	Data Structures	4	MTH 501	Probability and Statistics	3			
	Humanities and social III	5	CST_D 502	Formal Language and Automata	2			
	English language III	2	CST_D 503	Introduction to Artificial Intelligence	2			
	Physical Education III	1		Physical Education IV	1			
MTH 401	Introduction to Functions of One Complex Variable (Math/Sci Elec)	2		English language IV	2			
CST_D 402	Fundamentals of Electronics & Lab	4		Elective of humanities and social	1			
			CST_D 504	Introduction to High Performance Computing	2			
	Total Credit Hours	23		*Total Credit Hours*	18		*Total Credit Hours*	5

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 7			Semester 8			Semester 9		
CST_D 701	Computer Organization	4	CST_D 801	Operating System	3	CST_D 901	Design of Whole Computer System (Lab Course)	5
CST_D 702	Computer Network	3	CST_D 802	Computer Architecture	3			
CST_D 703	Principle of Compiler System	2	CST_D 803	Numerical analysis	3			
CST_D 704	Software Engineering	3		Politics Theory	4			
CST_D 705	Principle of Signal Processing	3		Humanities and social IV	3			
	Elective of humanities and social	2	CST_D 804	Emerging technology on Software (Tech Elective)	2			
CST_D 706	Introduction of VLSI Design (Tech Elective)	2						
	Total Credit Hours	19		*Total Credit Hours*	18		*Total Credit Hours*	5
Semester 10			Semester 11					
	Humanities and social V	5	CST_D 1101	Capstone project for graduation	15			
CST_D 1001	Network Security and Management (Tech Elective)	3						
CST_D 1002	Embedded System (Tech Elective)	3						
	Total Credit Hours	11		*Total Credit Hours*	15			

B.7.4 Mapping of Computer Engineering BoK to Curriculum D

Refer to section B.1.2 for an explanation of this table.

BoK Area / Course	CAE	CAL	CAO	DIG	ESY	NWK	PPP	SEC	SGP	SPE	SRM	SWD	ACF	DSC	LAL	PRS
Minimum Core BoK Hours	**50**	**30**	**60**	**50**	**40**	**20**	**20**	**20**	**30**	**35**	**20**	**45**	**30**	**30**	**20**	**30**
CST_D101												1-6				
CST_D102																
CST_D201												1,3,4				
CST_D301												3-8				
CST_D401		1-8,10														
CST_D402	1-10, 12															
CST_D501				1-10												
CST_D502#																
CST_D503#																
CST_D504			1-2, 10-11													
CST_D601			3,8,9													
CST_D602												4,8				
CST_D701			1-9	8												
CST_D702						1-10	6,7	8,9								
CST_D703		10														
CST_D704							3-5,10			1-5, 7-9						
CST_D705									1-8							
CST_D706												1-2, 9-10				
CST_D801											1-7					
CST_D802			1-2, 10-11													
CST_D803#																
CST_D804																
CST_D901										1-12						
CST_D1001						8		1-11								
CST_D1002					1-13											
CST_D1101							1-11									
MTH101													1-4			
MTH102															1-5	
MTH103														1-6		
MTH201													1-2, 5-7			
MTH202															1-2, 6-10	
MTH203														1-2, 7-9		
MTH401#																
MTH501																1-9
Core BoK Units Covered	**1-10**	**1-8**	**1-11**	**1-10**	**1-12**	**1-8**	**1-10**	**1-11**	**1-8**	**1-12**	**1-6**	**1-10**	**1-7**	**1-9**	**1-10**	**1-9**
Supplementary BoK Units	**12**	**10**			**13**	**9-10**	**11**				**7**					

 # this course does not cover any BoK Unit.

B.7.5 Curriculum D – Course Summaries

CST_D101 Fundamentals of Programming
Introduction to the basic concepts of computer programming, including problem solving, algorithmic thinking, simple data structure, and some fundamental algorithms
Prerequisite: None
Credit Hours: 3 Lecture Hours: 48; Lab Hours: 0
BoK Coverage: *CE-SWD 1-6*

CST_D102 Introduction of Information Science
Introductory course in information science; fundamental problems and solving methods including some emerging technologies in information science
Prerequisite: None
Credit Hours: 1; Lecture Hours: 16; Lab Hours: 0
BoK Coverage: CE-SET 1, 3-4

CST_D201 Fundamentals Object-oriented Programming
The basics of C++ programming, object-oriented programming, introductory problem solving, and computer programming using object-oriented techniques
Prerequisite: CST_D101
Credit Hours: 2; Lecture Hours: 32; Lab Hours: 0
BoK Coverage: CE-SWD 1-2, 6-8

CST_D301 Practice in Programming
Highly practical course aiming at students' innovative ideas and hands-on skills; through teaching and experiments, students develop general knowledge about C language and basic programming algorithms
Prerequisite: CST_D101
Credit Hours: 2; Lecture Hours: 16; Lab Hours: 32
BoK Coverage: CE-SWD 3-8

CST_D401 Data Structures
Data abstraction and representation, data structure (e.g. vector, array, list, stack, queue, tree, priority queue, hash table and graph) design and implementation, design of classic algorithms (e.g. search, sort and select, traverse, pattern matching, topological search, hash), as well as algorithm evaluation and analysis
Prerequisite: CST_D101
Credit Hours: 4; Lecture Hours: 64; Lab Hours: 32
BoK Coverage: CE-CAL 1-8, 10

CST_D402 Fundamentals of Electronics & Lab
Fundamentals of electric circuits, including the basic concepts (e.g. voltage, current, power), basic components (e.g. resistance, inductance, capacity, MOS transistor, operational amplifier), basic laws (e.g., Ohm's law, Kirchhoff's law, equivalent resistance exchange) and methods of circuit analysis
Prerequisite: None
Credit Hours: 4; Lecture Hours: 48; Lab Hours: 16
BoK Coverage: CE-CAE 1-10, 12

CST_D501 Digital Logic Circuits & Lab
Emphasis on the analysis, design, and implementation of the digital logic circuit; Boolean algebra and logic simplification, design and analysis of combinational circuits and sequential circuits, competition and risk management in logic design
Prerequisite: CST_D402
Credit Hours: 4; Lecture Hours: 48; Lab Hours: 16
BoK Coverage: CE-DIG 1-10

CST_D502 Formal Language and Automata

Focusing on formal languages, context free languages, and the related computation models as well as the Turing machine and the foundations of theoretical computing

Prerequisite: MTH103, MTH203
Credit Hours: 2; Lecture Hours: 32; Lab Hours: 0
BoK Coverage: *None*

CST_D503 Introduction to Artificial Intelligence

Introductory course on artificial intelligence; fundamental problems in artificial intelligence, production system, search problem, AND/OR search graph, predicate calculus, advanced search, expert systems

Prerequisite: CST_D101, MTH103, MTH203
Credit Hours: 2; Lecture Hours: 32; Lab Hours: 0
BoK Coverage: *None*

CST_D504 Introduction to High Performance Computing

Organization of high performance computer, design methods of parallel programming, performance model of programs, performance evaluation and optimization techniques, programming in MPI and OpenMP and algorithms in high performance computing

Prerequisite: CST_D101
Credit Hours: 2; Lecture Hours: 32; Lab Hours: 0
BoK Coverage: CE-CAO1-2, 10-11

CST_D601 Practice in Assembly Language

The basic theory, programming tools, programming methods and applications of assembly language

Prerequisite: CST_D101
Credit Hours: 3; Lecture Hours: 32; Lab Hours: 32
BoK Coverage: CE-CAO 3, 8-9

CST_D602 Practice in Java

Java programming, Java's history and evolution, development tools, object-oriented programming methods and network programming technologies

Prerequisite: CST_D101
Credit Hours: 2; Lecture Hours: 32; Lab Hours: 32
BoK Coverage: CE-SWD 4, 8

CST_D701 Computer Organization

Introduction to single CPU computer organization and architecture; instruction set architecture, datapath and controller, pipelining and performance measurement, hierarchical memory organization, bus structures, I/O interfacing and communications, peripheral equipment

Prerequisite: CST_D101, CST_D501
Credit Hours: 4; Lecture Hours: 48; Lab Hours: 16
BoK Coverage: CE-CAO 1-9, CE-DIG 8

CST_D702 Computer Network

Computer network architecture, typical protocols and standards of LAN, WAN, wireless and mobile networks, classical network applications, network management and security; networks social effects, network engineer's professional and responsibilities

Prerequisite: CST_D101, CST_D401
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage: CE-NWK 1-10, CE-PPP 6-7, CE-SEC 8-9

CST_D703 Principle of Compiler System

Overview of compilers, formal languages, grammar and automata, lexical analysis, syntax analysis, syntax-directed translation, semantic analysis and intermediate code generation, symbol table, run time storage organization, code optimization and generation

Prerequisite: CST_D401, CST_D502, CST_D602,
Credit Hours: 2; Lecture Hours: 32; Lab Hours: 0
BoK Coverage: None

CST_D704 Software Engineering

Focusing on the software development methods, software life cycle, design methods, software testing and performance evaluation, software development management

Prerequisite: CST_D101, CST_D401
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage: CE-SPE 1-5,7-9 CE-PPP 3-5,10

CST_D705 Principle of Signal Processing

Concepts on signal and signal processing, continuous-time Fourier transform, discrete-time Fourier transform, Z-transform and discrete time system analysis, design, and implementation of filters

Prerequisite: MTH101, MTH201
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage: CE-SGP 1-8

CST_D706 Introduction to Database System

Entity-relationship model and relational model; concepts, logic, and physical design of database, key techniques in database management system (e.g., query, transaction management, concurrency control and error recovery), expansion of database system structure, current research in database research, and new applications

Prerequisite: CST_D101, CST_D401
Credit Hours: 2; Lecture Hours: 32; Lab Hours: 0
BoK Coverage: CE-DIG 9-11

CST_D801 Operating System

Introduction to the computer operating system, process management, process scheduling, storage management, file management and device management

Prerequisite: CST_D101, CST_D401, CST_D701
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage: CE-SRM 1-7

CST_D802 Computer Architecture

Focusing on the structure, design principles, and other key concepts in computer architecture; design principles and performance evaluation of computer architecture, parallel technology in timing and space, multicore processor and multiprocessor system, high performance computing and networks, memory hierarchy architecture for single core and multicore systems

Prerequisite: CST_D501, CST_D701, CST_D703
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage: CE-CAO 1-2, 10-11 CE-SET 4

CST_D803 Numerical analysis

Numerical computation and error analysis, direct and iteration methods to solve system of linear equations, calculate Eigen values and Eigen vectors of matrices; function approximation and interpolation method: Lagrange's interpolation, Newton's interpolation, subsection low interpolation, and cubic spline interpolation; approximation for numerical integration and differential; approximation method for solving equations

Prerequisite: MTH101, MTH102, MTH201, MTH202
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage:

CST_D804 Emerging technology on Software

Introduction to the frontier researches of software technology, including system software, data and knowledge engineering, software engineering, computer aided design technology

Prerequisite:
Credit Hours: 2; Lecture Hours: 32; Lab Hours: 0
BoK Coverage: CE-SET 1-5

CST_D901 Design of Whole Computer System

Practice course on the design and implement of whole computer system—two or three students working together to design and implement a mini-computer with the basic hardware, operating system, and complier

Prerequisite: CST_D701, CST_D702, CST_D703, CST_D801
Credit Hours: 5; Lecture Hours: 20; Lab Hours: 140[3]
BoK Coverage: CE-SPE 1-12

CST_D1001 Information Security and Network Management

Foundations of cryptography, symmetric cryptography, public key cryptography, Hash algorithms, digital signature, user authentication technology, network security techniques, structure, and protocol standards for network management and operations

Prerequisite: CST_D702
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage: CE-SEC 1-11, CE-NWK 8

CST_D1002 Embedded System

Components of embedded system (e.g., embedded processor; flash embedded memory; bus and communication in embedded systems, sensors, and drivers), power source design, boot process, real time operating systems, driver design, mid-ware and other embedded software, development, and test methods

Prerequisite: CST_D701, CST_D801
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage: CE-ESY 1-13

CST_D1101 Capstone project for graduation

The capstone project for graduation to ensure that all graduates have significant design experience, as well as experience with teamwork and modern engineering tools

Prerequisite: None
Credit Hours: 15; Lecture Hours: 32; Lab Hours: 400[4]
BoK Coverage: CE-PPP 1-11

MTH_D101: Calculus I

Introduction to the basic knowledge of limit theorem, including functional limit, continuity of function, and the calculation of limit; includes calculus methods such as derivative and differential, integral, ordinary differential equation and Improper integrals; series, convergence criteria, convergence domain and uniform convergence of function series, term-by-term limit, term-by-term summation, term-by-term integral

Prerequisite: None
Credit Hours: 5; Lecture Hours: 80; Lab Hours: 0
BoK Coverage: CE-ACF 1-4

MTH_D102: Geometry and Algebra I

One of the basic math courses in higher education, introducing the fundamental knowledge related to geometry, algebra, and the relationship between these two fields; cover mapping, geometry order, and basic concepts of group, ring, and field, as well as vectors in geometry spaces, linear space, and inner product space; algebra-related contents including linear mapping, matrix, concept and properties of determinant, linear equation system, orthogonal matrixes and similar matrixes

Prerequisite: None
Credit Hours: 4; Lecture Hours: 64; Lab Hours: 0
BoK Coverage: CE-LAL 1-5

MTH_D103: Discrete Mathematics I

Logic proposition and propositional calculus, first order predicate logic and predicate calculus, natural formal system of reasoning in propositional calculus and predicate calculus, as well as, operation and property of set and binary relation, functions on set and their properties

Prerequisite: None
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage: CE-DSC 1-6

[3] Summer assignment. Students must design and implement their own complete computer within five weeks; no other task occurs during this summer session.
[4] In China, undergraduate students must do a capstone project in their last semester that lasts at least 16 weeks and it has 400 lab hours.

MTH$_D$201: Calculus II

Knowledge related to multivariate functions; derivative, integral, line/surface integrals of the first type, line/surface integrals of the second type, plane vector field and Green formula, space vector field and Gauss formula, Stokes formula, path independent integral, linear ordinary differential equation of second order and system of linear ordinary differential equation of first order

Prerequisite: MTH$_D$101
Credit Hours: 5; Lecture Hours: 80; Lab Hours: 0
BoK Coverage: CE-ACF 1-2, 5-7

MTH$_D$202: Geometry and Algebra II

Quadratic forms, quadratic curve and its categorization, common curved surface, space curve equation, quadratic surface and its categorizations, as well as, plane orthogonal transformation, plane affine transformation, projection plane and homogeneous coordinates, projective transformation and projective mapping, vector function and its calculus, arc length of a curve and Frenet frame

Prerequisite: MTH$_D$102
Credit Hours: 2; Lecture Hours: 32; Lab Hours: 0
BoK Coverage: CE-LAL 1-2, 6-10

MTH$_D$203: Discrete Mathematics II

Graphing and its algebraic representation; path, cycle, and connectivity, as well as, tree, algebraic structure, group, ring, field, lattice, and Boolean algebra

Prerequisite: MTH$_D$103
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage: CE-DSC 1-2, 7-9

MTH$_D$501: Probability and Statistics

Mathematical descriptions of random phenomenon, focusing on the core concepts of probability and statistics; probability space, determining the probability of random events, conditional probability and its applications in probability calculation, statistical independence of random events, random variable and its distribution, quantitative statistical analysis of random phenomenon, basic methods of statistical analysis and statistical inference using probability models

Prerequisite: None
Credit Hours: 3; Lecture Hours: 48; Lab Hours: 0
BoK Coverage: CE-PRS 1-9

B.8 Curriculum E: Bologna-3 Model

Computer Engineering Program Representative of a First Cycle Program compatible with the Bologna Declaration

B.8.1 Program Goals and Features

This curriculum model demonstrates a typical program in computer engineering as one might find in first cycle programs within the European Higher Education Area, the area within which the Bologna arrangements operate.

The Bologna Declaration was signed on the 19th of June 1999 in Bologna, Italy, by ministers from some 29 countries in charge of higher education. Its purpose was to create a European Higher Education Area (EHEA) that would facilitate the mobility of students between countries and enhance the quality of educational provision [EHEA]. Essentially, they saw degree structures taking the form of a first cycle bachelor's degree taking three years and a second cycle master's degree taking a further two years, or five years in total. Accompanying these proposals were requirements about standardizing the numbers of credits within degrees and ensuring comparability across institutions in the work required of students. As of May 2015, 47 countries and over 4000 higher education institutions have adopted the Bologna arrangements.

The basis for comparability within Bologna is the European Credit Transfer System (ECTS) and the ECTS credit. In brief, one academic year counts as 60 ECTS credits, equivalent to 1500-1800 hours of study in all countries. Typically, a bachelor's degree requires 180-240 ECTS credits taking 3-4 years, and a master's degree 90-120 ECTS credits taking 1-2 years. The related qualifications frameworks depend on learning outcomes; they have become essential to the Bologna process.

The Bologna framework attempts to connect different educational systems rather than harmonizing them. Considerable flexibility exists in the Bologna arrangements to facilitate widespread compatibility. The UK, for instance, has adopted the Bologna arrangements and yet the idea of an honors degree (not present within Bologna) is dominant, different (but compatible) credit systems and qualification frameworks are often used, and master's degree programs often take (a full) 12 months.

The Bologna arrangements accommodate a wide variety of degree programs, many arrangements including elements such as cultural issues and language studies often for facilitating mobility between countries. This curriculum example offers guidance on computer engineering degrees for the first cycle, seen essentially as providing the foundations of computer engineering. Second cycle arrangements are typically more advanced and they tend to focus on specialization as would be expected of master's degree programs.

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 that 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. And they will also be equipped to move to a related second cycle or master's degree program. Graduates must possess design skills and will have the capacity to apply their accumulated 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.

B.8.2 Summary of Requirements

This three-year program assumes that students have completed two semesters of calculus and two semesters of physics prior to entering the program. These are indicated as prerequisites to various classes in this model.

The introduction of concepts in computer engineering appears in the early years. The justification for this is that students should sample the discipline they will study, a matter deemed important for motivational purposes. For the third year, the curriculum includes several optional classes.

In these programs of study, some element of laboratory experience constitutes 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 will 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 three-year illustration of a computer engineering program contains 30 courses and 90 credit hours of study. The distribution of credit hours is as follows.

Credit-hours	*Areas*
15*	Mathematics
0*	Natural science (physics, chemistry)
0	Humanities, social sciences, composition, literature
21	Required computer engineering (CE)
21	Required electrical and computer engineering (ECE)
9	Required software engineering
12	Required computer science (CSC)
6	Technical electives
6	Design project (CE)
0	Free electives
*90**	*TOTAL Credit Hours for Computer Engineering Program Bologna-3.*

* Does not include prerequisite credit hours in calculus, physics, and other related college-level courses taken prior to entering the program.

The illustration allows for electives (options) available in the third year of study. Students may choose two third-year electives selected from a set of four computer engineering courses (modules). These four third-year electives are: Computer Graphics, Intelligent Systems and Robotics, Device Development, and Multimedia Systems.

What follows is a representation of a possible curriculum models for a three-year degree program. Following this is a mapping of courses from the program to the body of knowledge for computer engineering. The courses (or modules) appear in the Course Summaries, which follows the illustration of the computer engineering curriculum.

B.8.3 Three-Year Model for Curriculum E

Course	***Description***	***Credit***	***Course***	***Description***	***Credit***
Semester 1			**Semester 2**		
MTH_E101	Discrete Structures for Computing	3	MTH_E103	Calculus and Geometry	3
MTH_E102	Applied Probability and Statistics	3	MTH_E104	Linear Algebra	3
CSC_E101	Computer and Information Systems	3	CE_E101	Concepts in Computer Engineering	3
ELE_E101	Foundations of Electronics	3	ELE_E102	Digital Circuits I	3
SWE_E101	Programming Basics	3	SWE_E102	Programming Fundamentals	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 3			**Semester 4**		
MTH_E201	Mathematics for Engineers	3	CSC_E202	Operating Systems and Net-Centric Computing	3
CE_E201	Networking & Communications	3	CE_E203	Professional Issues in Computer Engineering	3
CSC_E201	Analysis and Design of Algorithms	3	ELE_E202	Analogue Circuits	3
CE_E202	Computer Organization	3	CE_E 204	Computer Systems Engineering	3
ELE_E201	Digital Circuits II	3	CE_E205	Embedded Computer Systems	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*
Semester 5			**Semester 6**		
CE_E301	Individual Project I	3	CE_E302	Individual Project II	3
CSC_E301	Programming Languages and Syntax Directed Tools	3	CE_E303	Computer Architecture	3
ELE_E301	Signals and Systems	3	ELE_E302	System Control	3
SWE_E301	Software Engineering	3	ELE_E303	Digital Signal Processing	3
	Technical Elective (Option A)	3		Technical Elective (Option B)	3
	Total Credit Hours	*15*		*Total Credit Hours*	*15*

B.8.4 Mapping of Computer Engineering BoK to Curriculum E

Refer to section B.1.2 for an explanation of this table.

BoK Area / Course	CAE	CAL	CAO	DIG	ESY	NWK	PPP	SEC	SGP	SPE	SRM	SWD	ACF	DSC	LAL	PRS
Minimum Core BoK Hours	**50**	**30**	**60**	**50**	**40**	**20**	**20**	**20**	**30**	**35**	**20**	**45**	**30**	**30**	**20**	**30**
CE_E101								1-5		1-5						
CE_E201						1-8										
CE_E202			1-7													
CE_E203							1-10									
CE_E204								6-11		6-10						
CE_E205					1-12											
CE_E303			8-11													
CSC_E101		1-3				1-3				1-4	1-3					
CSC_E201		1-10														
CSC_E202						1-3		1-4			1-6					
CSC_E301			2-5									4-10, 12				
ELE_E101	1-6															
ELE_E102				1-7												
ELE_E201	7-10			8-10												
ELE_E202	4-8															
ELE_E301									1-8							
ELE_E302				7-10												
ELE_E303									7-8							
MTH_E101														1-9		
MTH_E102																1-9
MTH_E103													1-7			
MTH_E104															1-10	
MTH_E201																
SWE_E101												1-7				
SWE_E102		1-8														
SWE_E301												8-12				
Core BoK Units Covered	**1-10**	**1-8**	**1-11**	**1-10**	**1-12**	**1-8**	**1-10**	**1-11**	**1-8**	**1-12**	**1-6**	**1-10**	**1-7**	**1-9**	**1-10**	**1-9**
Supplementary BoK Units	**11-12**	**10**		**11**	**13**	**9-11**	**11**		**9-10**		**7-8**	**11-12**				**10**

B.8.5 Curriculum E – Course Summaries

CE_E101: 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 (achieved before entering the program)
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-SEC 1-5, CE-SPE 1-5

CE_E201: 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 (achieved before entering the program)
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-NWK 1-8

CE_E202: 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: CE_E101
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-CAO 1-7

CE_E203: 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; process of ethical decision-making, privacy and confidentiality, computer crime, professional codes and responsibilities, professional practice, system security and more generally cyber security, impact of computers on society

Prerequisites: Second-year standing
Credit Hours 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-PPP 1-10

CE_E204: 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: CE_E101
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 28
BoK Coverage: CE-SEC 6-10, CE-SPE 6-10

CE_E205: Embedded Computer Systems

Nature of embedded systems, specific 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
BoK Coverage: CE-ESY 1-12

CE_E301: Individual Project I

Comprehensive individual project spanning two semesters, addressing a significant technical problem under the guidance of a supervisor; students 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: Third-year standing
Credit Hours: 3; Lecture Hours 42; Lab Hours: 14
BoK Coverage: Project dependent

CE_E302: Individual Project II

Continuation of Individual Project I

Prerequisites: Third-year standing
Credit Hours: 3; Lecture Hours: 0; Lab Hours: 42
BoK Coverage: Project dependent

CE_E303: 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: CE_E202
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-CAO 8-11

CSC_E101: 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: First year standing
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-NWK 1-3, CE-CAL 1-3, CE-SPE 1-4, CE-SRM 1-3

CSC_E201: 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_E101, SWE_E101
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 28
BoK Coverage: CE-CAL 1-10

CSC_E202: 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; high-level languages, real-time issues, networking, security, multimedia; file systems, hierarchical design; process management, scheduling strategies; resource allocation strategies; concurrency, synchronization principles, deadlock avoidance; device drivers and interfacing; net centric computing; considerations about different platforms and mobility

Prerequisites: CSC_E101
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-NWK 1-3, CE-SEC 1-4, CE-SRM 1-6

CSC_E301: 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 (achieved before entering the program)
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 28
BoK Coverage: CE-SWD 4-10, 12

ELE_E101: 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; electrical properties of materials; diodes and diode circuits; MOS transistors and biasing, MOS logic families

Prerequisites: Two courses in calculus and two courses in physics (achieved before entering the program)
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-CAE 1-6

ELE_E102: 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 (achieved before entering the program)
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-DIG 1-7

ELE_E201: 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_E102
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 28
BoK Coverage: CE-DIG 8-10, CE-CAE 7-10

ELE_E202: 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_E201
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-CAE 4-8

ELE_E301: 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: ELE_E201, ELE_E202
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-SGP 1-6

ELE_E302: 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

Prerequisites: ELE_E301
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-DIG 7-10

ELE_E303: 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_E301
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-SGP 7-8

MTH_E101: 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 (achieved before entering the program)
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 0
BoK Coverage: CE-DSC 1-9

MTH_E102: 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; packages supporting data analysis

Prerequisites: Two courses in calculus (achieved before entering the program)
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-PRS 1-9

MTH_E103: Calculus and Geometry

Review of basic differential and integral calculus; techniques and approaches; single and double integrals; simple differential equations and their solutions; complex numbers; vector calculus; graphical concepts supported by appropriate graphics packages

Prerequisites: Two courses in calculus (achieved before entering the program)
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-ACF 1-7

MTH_E104: Linear Algebra

Bases, vector spaces and orthogonality; matrix representation of linear systems; matrix inversion; linear transformations; solution of linear systems; solution of non-linear systems; determinants; eigenvectors and eigenvalues; use of appropriate packages for linear algebra

Prerequisites: MTH_E101
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: CE-LAL 1-10

MTH_E201: Mathematics for Engineers

Introduction to numerical methods and their uses in engineering; simulation and modeling: basic principles and techniques; applications in engineering; use of tools in support of engineering simulation and modeling; queuing theory including system simulation and modeling, queuing methods; use of appropriate statistical packages; complex numbers; Fourier transforms

Prerequisites: MTH_E102, MTH_E103, MTH_E104
Credit Hours: 3; Lecture Hours: 42; Lab Hours: 14
BoK Coverage: Not applicable

SWE_E101: 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: First year standing
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 28
BoK Coverage: CE-SWD 1-7

SWE_E102: 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_E101
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 28
BoK Coverage: CE-CAL 1-8

SWE_E301: 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: SWE_E101, SWE_E102
Credit Hours: 3; Lecture Hours: 28; Lab Hours: 28
BoK Coverage: CE-SWD 8-12

Appendix C

Computer Engineering Laboratories

This appendix to the *Computing Curricula - Computer Engineering (CE2016)* report describes possible laboratory configurations useful for developing modern student laboratory experiences for computer engineering programs. This appendix reflects the discussion presented in sections 4.4 and 6.3.3 of this report. The steering committee does not endorse any product or manufacturer. The items listed here only serve as a guide in developing student experiences in a laboratory environment for computer engineering programs.

C.1 Circuits and Electronics

Typical Description: Experimental use of laboratory instruments; voltage, current, impedance, frequency, and waveform measurements; elements of circuit modeling and design; design, construction, and simulation of filters; components of periodic signals.

Typical Configuration:
A one- or two-student workstation includes:

- platform/breadboard for circuit construction
- triple-output DC power supply
- two-channel mixed-signal oscilloscope
- multimeter
- function/arbitrary waveform generator
- computer with circuit-level modeling and simulation tools and instrumentation control

The test instruments may be standalone, integrated into the platform/breadboard, or personal instrumentation owned by the department or the students and used with personal computers.

Vendors for this equipment in 2016 include Agilent Technologies, National Instruments, Tektronix, Fluke, Hewlett-Packard, and others.

Typical Offering: Lower level; one three-hour laboratory experience per week.

C.2 Computer Architecture Design

Typical Description: Techniques of design, simulation, and evaluation of a simple datapath and control using a hardware description language (e.g., VHDL or Verilog); assembly language programming on an emulated 32- or 64-bit microprocessor; implementation of an RTL model of an instruction set architecture in an FPGA.

Typical Configuration:

- computer with VHDL and/or Verilog modeling and simulation tools
- FPGA development board
- FPGA development suite to support the selected FPGA development board
- oscilloscope and logic analyzer to examine FPGA board outputs

Typical Offering: Lower level; one two-hour laboratory experience per week.

C.3 Digital Logic Design

Typical Description: Experiments involving digital circuits of increasing complexity; combinational small-scale integration (SSI) and medium-scale integration (MSI) circuits; arithmetic and sequential circuits; analysis and synthesis of state machines.

Typical Configuration:

- breadboard for constructing digital circuits from SSI/MSI components
- oscilloscope and logic analyzer (separate instruments or integrated into the breadboard)
- FPGA development board
- computer with VHDL/Verilog modeling and simulation tools and FPGA development suite

Typical Offering: Lower level; one three-hour laboratory experience per week.

C.4 Digital Signal Processing

Typical Description: Engage in hardware and software experiments showing digital signal processing principles and techniques; programming on DSP chips; real-time signal processing algorithms.

Typical Configuration:

- digital signal processor (DSP) development board/kit
- computer with DSP software development tools
- mixed-signal oscilloscope
- logic analyzer
- vector signal generator
- spectrum analyzer

Vendors for this equipment in 2016 include Texas Instruments, ARM, Tektronix, Agilent, Rhode & Schwarz, and others.

Typical Offering: Upper level; one three-hour laboratory experience per week.

C.5 Digital Logic and System Design

Typical Description: Hierarchical, modular design of digital systems of increasing complexity; design, analysis, and synthesis of state machines; computer-aided digital system modeling, simulation, analysis, and synthesis; design implementation with programmable logic devices and/or FPGAs.

Typical Configuration:

- computer to host design tools
- VHDL and/or Verilog modeling and simulation tools
- FPGA development board
- proto board for interfacing peripheral components with the FPGA
- FPGA development suite to support the selected FPGA development board
- embedded processor soft core, or FPGA with an embedded processor hard core
- oscilloscope
- logic analyzer
- test pattern generator to provide FPGA inputs

Vendors for this equipment in 2016 include Digilent, Xilinx, Aldec, Altera, Tektronix, and others.

Typical Offering: Upper level; one three-hour laboratory experience per week.

C.6 Embedded Systems

Typical Description: Experiments involving interfacing memory and peripheral devices to a microcomputer; design of software to control peripheral devices; integration of computer hardware and software for system control.

Typical Configuration:

- microcontroller development board/kit
- computer for hosting software tools
- integrated development environment for the selected microcontroller
- libraries of software modules to support selected peripheral devices
- powered breadboard for interfacing peripheral devices to the microcontroller
- oscilloscope
- logic analyzer
- multimeter
- triple output DC power supply

Vendors for this equipment in 2016 include ST Microelectronics, Digilent, Keil, Agilent, and others.

Typical Offering: Lower level; one two-hour laboratory experience per week.

C.7 Engineering Introduction

Typical Description: Basic engineering skills and practice; students learn the basics of circuits, DC motors, and wireless communication; a design project involving some aspect of computer engineering (e.g., a radio-controlled car) culminates the experience; focus is on engineering design, teamwork, communication skills, and other related activities. This is often taught as a multi-disciplinary course.

Typical Configuration: Varies, based on the design project and orientation of the course.

Typical Offering: Lower level; one two-hour laboratory experience per week.

C.8 Networking

Typical Description: Design and implementation of information networks based on requirements and devices such as routers and switches; applications of information networks for data, audio, and video communications; transmission media, modulation, error control, flow control, LANs, and Ethernet protocols; experiments on data communication signaling and error control; data transfer and software aspects of networks common in computing; implementation of servers and clients using various protocols.

Typical Configuration:

- multiple personal computers, to be integrated into a network
- network isolated from the institutional network
- configurable routers and/or switches
- network analyzer and/or software tools to measure network traffic and conditions

Vendors for this equipment in 2016 include Emona, Cisco, and others.

Typical Offering: Upper level; one three-hour laboratory experience per week.

C.9 Software Design

Typical Description: Experience in software construction; testing, debugging, and associated tools; configuration management; low-level file and device I/O; systems and event-driven programming; languages to include C, C++, C#, Python, Ruby, Java and/or other appropriate languages in support of the computer engineering program.

Typical Configuration:

- modern computing platforms running widely-used modern operating systems
- IDE to manage project files and libraries
- compiler(s) and linker for language(s) used in the course
- source-level debugger
- documentation, presentation, file transfer, and other support tools
- mathematics package for analysis, simulation, and modeling
- database software

Typical Offering: Lower level; one two-hour laboratory experience per week.

Appendix D

Acknowledgements and Dissemination

Throughout the development of this report, comments and suggestions were solicited from the global computer engineering community. This was accomplished through two principal methods.

First, members of the steering committee and others associated with the development effort presented papers, panel sessions, and workshops at numerous conferences. Workshop participants provided specific feedback, while attendees in the other sessions were encouraged to provide feedback at that time or by contacting committee members. It is estimated that collectively over five hundred people attended sessions at the events shown in Table D.1 and many have provided input to the report.

Two draft versions of the report were posted online with a web-based form for individuals to provide comments. Email invitations to review the document were sent to members of relevant professional societies, including ACM SIGCSE, the IEEE Education Society, and other organization lists.

The steering committee is very appreciative of all comments and suggestions received during this process. When providing comments and suggestions, the following individuals provided their names and so are specifically acknowledged here.

Jose L. Aguilar C., Universidad de Los Andes, Venezuela and Ecuador
Xiaoying Bai, Tsinghua University, Beijing, China
Olga I. Bogoiavlenskaia, Petrozavodsk State University, Karelia, Russia
Iurii A. Bogoiavlenskii, Petrozavodsk State University, Karelia, Russia
Tarek El-Bawab, IEEE Communications Society, USA
Manuel Gericota, School of Engineering-Polytechnic of Porto, Portugal
Jorge Guerra, Universidad Nacional Mayor de San Marcos (UNMSM), Peru
Wilfrido Inchaustti, Universidad Nacional Mayor de San Marcos (UNMSM), Paraguay
Qin Leihua, Huazhong University of Science and Technology, Wuhan, China
Zhang Liang, Fudan University, Shanghai, China
Doug Lyon, Fairfield University, Fairfield, Connecticut, USA
Clive Maynard, Curtin University, Perth, Australia
Doug Myers, Curtin University, Perth, Australia
Richard Perry, Villanova University, Villanova, Pennsylvania, USA
Carlos Ribeiro, Instituto Superior Técnico, Universidade de Lisboa, Portugal
Cristian Rusu, Pontificia Universidad Catolica de Valparaiso, Chile
Mitch Thornton, Southern Methodist University, Dallas, Texas, USA
Murali Varanasi, University of North Texas, Denton, Texas, USA
Timothy Wilson, Embry-Riddle Aeronautical University, Daytona Beach, Florida, USA
Tang Yuhua, National University of Defense Technology, Changsha, China
Wang Zhiying, National University of Defense Technology, Changsha, China

The CE2016 steering committee thanks these individuals for their comments and suggestions in the development of this report.

Table D.1. CE2016 Presentation Events

Year	Dates	Event	Location
2012	February 29-March 3	43rd ACM Technical Symposium on Computer Science Education	Raleigh, North Carolina, USA
2012	October 3-6	2012 Frontiers in Education (FIE) Conference	Seattle, Washington USA
2013	June 23-26	2013 ASEE Annual Conference and exposition	Atlanta, Georgia, USA
2013	October 23-26	2013 Frontiers in Education (FIE) Conference	Oklahoma City, Oklahoma, USA
2014	June 15-18	2014 ASEE Annual Conference and exposition	Indianapolis, Indiana, USA
2014	October 22-25	2014 Frontiers in Education (FIE) Conference	Madrid, Spain
2014	November 3-7	IV Simpósio Brasileiro de Engenharia de Sistemas Computacionais	Manaus, AM, Brazil
2014	December 8-10	IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE 2014)	Wellington, New Zealand
2015	March 18-20	IEEE EDUCON Global Engineering Education Conference (EduCon 2015)	Tallinn, Estonia
2015	April 23-24	13th China International Software Cooperation Conference (CHINASOFT 2015)	Chengdu, China
2015	April 25	Sichuan University	Chengdu, China
2015	April 25-26	Beijing Computer Education Research Association	Beijing, China
2015	June 14-17	2015 ASEE Annual Conference and exposition	Seattle, Washington, USA
2015	October 21-24	2015 Frontiers in Education (FIE) Conference	El Paso, Texas, USA
2015	November 27-29	University Fundamental Course Forum (UFCF)	Tianjin, China
2015	December 1-2	Xi'an Jiaotong University	Xi'an, China
2016	March 18-22	2016 Electrical and Computer Engineering Department Heads Association (ECEDHA) Annual Conference	La Jolla, California, USA
2016	October 12-15	2016 Frontiers in Education (FIE) Conference	Erie, Pennsylvania USA

References

[ABET, 2016] ABET Evaluation Criteria; http://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2016-2017/.

[ACM, 1992] ACM Code of Ethics and Professional Conduct, adopted 16 October 1992; https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct.

[ACM/IEEECS, 1999] ACM and IEEE Computer Society, Software Engineering Code of Ethics and Professional Practice, 1999; https://www.computer.org/cms/Publications/code-of-ethics.pdf.

[ACM/IEEECS, 2008a] Computer Science Curriculum 2008: An Interim Revision of CS2001; http://www.acm.org/education/CS2008.pdf

[ACM/IEEECS, 2008b] Information Technology 2008: Curriculum Guidelines for Undergraduate Degree Programs in Information Technology, November 2008; http://www.acm.org//education/curricula/IT2008%20Curriculum.pdf.

[ACM/IEEECS, 2013] Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, December 20, 2013; http://www.acm.org/education/CS2013-final-report.pdf.

[ACM/IEEECS, 2015] Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, February 23, 2015; http://www.acm.org/binaries/content/assets/education/se2014.pdf.

[ACM/AIS, 2010] IS 2010: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems, 2010; http://www.acm.org/education/curricula/IS%202010%20ACM%20final.pdf.

[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] Association of Information Technology Professionals (AITP), Code of Ethics and Standards of Conduct; http://c.ymcdn.com/sites/www.aitp.org/resource/resmgr/forms/code_of_ethics.pdf.

[AITP, 2002] Association of Information Technology Professionals, Code of Ethics, 2002; http://www.aitp.org/?page=EthicsConduct.

[ALIBAB] Alibabaoglan; http://www.alibabaoglan.com/blog/gartners-top-predictions-till-2020/.

[Bloom 1956] B.S. Bloom, ed., *Taxonomy of Educational Objectives: The Classification of Educational Goals: Handbook I, Cognitive Domain*, Longmans, 1956.

[BLS] Bureau of Labor Statistics; http://www.bls.gov/emp/ep_table_102.htm.

[CHIN] China Gorman; http://chinagorman.com/2013/04/16/you-think-we-have-skills-shortages-now-lets-talk-in-2020/.

[CC91] Computing Curricula 1991; http://dl.acm.org/citation.cfm?doid=103701.103710.

[DICT] Dictionary.com; http://dictionary.reference.com/browse/soft+skills.

[Dublin] Dublin Accord; http://www.ieagreements.org/dublin/.

[EHEA] European Commission/EACEA/Eurydice, The European Higher Education Area in 2015: Bologna Process Implementation Report, Luxembourg: Publications Office of the European Union.

[EngC] Engineering Council; www.engc.org.uk.

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

[IEEE] IEEE, Code of Ethics; http://www.ieee.org/about/corporate/governance/p7-8.html.

[IEEE, 1990] IEEE Code of Ethics, August 1990; http://ewh.ieee.org/cmte/substations/posted_documents/ieee_codeofethics.pdf.

[IFIP, 1998] Harmonization of Professional Standards (Draft Version), October 1998; http://www.ifip.org/minutes/C99/C99_harmonization.htm.

[IRPE] International Register of Professional Engineers, http://ncees.org/records/international-registry/.

[ITEEA] International Technology and Engineering Educators Association; http://www.iteea.org/.

[INVEST] Investopedia; http://www.investopedia.com/terms/s/soft-skills.asp.

[NSPE, 2003] National Society of Professional Engineers, NSPE Code of Ethics for Engineers, 2003; https://www.nspe.org/resources/ethics/code-ethics.

[OVERVIEW] Computing Curricula 2005, The Overview Report; http://www.acm.org/education/education/curric_vols/CC2005-March06Final.pdf

[SBS] Subject benchmark statement: Masters degrees in Computing 2011, The Quality Assurance Agency for Higher Education.

[SEEPP] Software Engineering Ethics and Professional Practices (SEEPP), Code of Ethics; https://www.acm.org/about/se-code.

[Seoul] Seoul Accord; http://www.seoulaccord.org/.

[SIGCAS] Special Interest Group on Computers and Society (SIGCAS); (ACM) http://www.sigcas.org/.

[SSIT] Society on Social Implications of Technology (SSIT) of IEEE; http://ieeessit.org/.

[Sydney] Sydney Accord; http://www.ieagreements.org/sydney/.

[Washington] Washington Accord; http://www.ieagreements.org/Washington-Accord.

Association for
Computing Machinery