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

Version

2015 October 25
Computer Engineering

Curricula 2016

Interim Report

2015 October

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

ACM Delegation

John Impagliazzo (Chair)
Hofstra University, USA

Susan Conry
Clarkson University, USA

Joseph L.A. Hughes
Georgia Institute of Technology, USA

Liu Weidong
Tsinghua University, China

Lu Junlin
Peking University, China

Andrew McGettrick
University of Strathclyde, Scotland

Victor Nelson
Auburn University, USA

IEEE Computer Society Delegation

Eric Durant
Milwaukee School of Engineering, USA

Lorraine Herger
IBM Research, USA

Herman Lam
University of Florida, USA

Robert Reese
Mississippi State University, USA
# Contents

CE2016 Steering Committee ........................................................................................................... 4  
Contents ........................................................................................................................................... 5  
Executive Summary .......................................................................................................................... 9  
(Some) Contributing Reviewers ....................................................................................................... 11  
Chapter 1: Introduction .................................................................................................................. 12  
1.1 Overall Structure of the Computing Curricula Project ............................................................... 12  
1.2 Overview of the CE2016 Process ................................................................................................. 13  
1.3 Underlying Principles ............................................................................................................. 13  
1.4 Structure of the CE2016 Report ............................................................................................... 14  
Chapter 2: Computer Engineering as a Discipline ......................................................................... 15  
2.1 Background .................................................................................................................................. 15  
2.2 Evolution of the field ............................................................................................................... 15  
2.3 Characteristics of computer engineering graduates ..................................................................... 17  
  2.3.1 Distinctions ........................................................................................................................... 17  
  2.3.2 Professionalism .................................................................................................................... 17  
  2.3.3 Ability to design .................................................................................................................. 18  
  2.3.4 Breadth of knowledge ........................................................................................................ 18  
2.4 Organizational considerations .................................................................................................. 19  
2.5 Preparation for professional practice ....................................................................................... 19  
2.6 Program evaluation and accreditation ...................................................................................... 20  
Chapter 3: The Computing Engineering Body of Knowledge ....................................................... 22  
3.1 Structure of the body of knowledge .......................................................................................... 22  
  3.1.1 Core and supplementary components .................................................................................. 22  
  3.1.2 Assessing the time required to cover a unit ......................................................................... 22  
  3.1.3 Tags for KAs and KUs .......................................................................................................... 23  
3.2 Learning Outcomes .................................................................................................................. 23  
3.3 Summary of the CE body of knowledge ................................................................................. 23  
  3.3.1 Related mathematics .......................................................................................................... 26  
  3.3.2 Related science .................................................................................................................... 26  
3.4 CE2016 BoK compared with CE2004 BoK ........................................................................... 26  
3.5 Rationale for 420 hours in computer engineering .................................................................... 27  
3.6 Curricular models ..................................................................................................................... 28  
Chapter 4: Engineering Practice and the Computer Engineering Curriculum ............................... 29  
4.1 The nature of computer engineering ....................................................................................... 29  
4.2 Design in the curriculum ......................................................................................................... 30  
  4.2.1 Design throughout the curriculum ..................................................................................... 30  
  4.2.2 The culminating design experience .................................................................................... 30  
4.3 The laboratory experience ....................................................................................................... 31  
4.4 The role of engineering tools .................................................................................................. 31  
4.5 Applications of computer engineering principles ..................................................................... 32  
4.6 Complementary skills ............................................................................................................. 32  
  4.6.1 Communication skills ......................................................................................................... 33  
  4.6.2 Teamwork skills ................................................................................................................ 34
4.6.3 Soft or personal skills................................................................. 34
4.6.4 Experience................................................................. 34
4.6.5 Lifelong learning................................................................. 35
4.6.6 Business perspectives................................................................. 35
4.7 Becoming professionals................................................................. 36
4.8 The elements of an engineering education................................................................. 36

Chapter 5: Professional Practice ................................................................. 38
5.1 Overview of professional practice................................................................. 38
5.1.1 Professional practice and the CE curriculum................................................................. 38
5.1.2 Professional needs................................................................. 39
5.2 Decisions in a Societal Context................................................................. 39
5.3 Professionalism and education................................................................. 40
5.3.1 Special student experiences ................................................................. 40
5.3.2 Administration, faculty, and student roles................................................................. 41
5.3.3 Incorporating Professional Practice into the Curriculum................................................................. 41
5.3.4 Professionalism and student experiences................................................................. 42
5.4 Professionalism and the workplace................................................................. 42
5.4.1 Private and public sectors ................................................................. 42
5.4.2 Modelling local and international work environments................................................................. 43
5.4.3 Certifications................................................................. 44
5.5 Fostering Professionalism................................................................. 44
5.5.1 Professional ethical codes ................................................................. 44
5.5.2 Education and professional practice................................................................. 45

Chapter 6: Curriculum Implementation Issues ................................................................. 46
6.1 General Considerations................................................................. 46
6.2 Principles for Curriculum Design................................................................. 47
6.3 Basic Computer Engineering Curriculum Components................................................................. 47
6.3.1 Coverage of the BOK Core................................................................. 47
6.3.2 Course Arrangement................................................................. 48
6.3.3 Lab Experiences................................................................. 48
6.3.4 Culminating Project................................................................. 48
6.3.5 Engineering Professional Practice................................................................. 48
6.3.6 Communication Skills................................................................. 48
6.4 Course Material Presented by Other Departments................................................................. 49
6.4.1 Mathematics Requirements................................................................. 49
6.4.2 Science Requirements................................................................. 50
6.4.3 Other Requirements................................................................. 50
6.5 Sample Curricula................................................................. 50

Chapter 7: Institutional Adaptations ................................................................. 52
7.1 The need for local adaptation................................................................. 52
7.2 Attracting and retaining faculty................................................................. 52
7.3 The need for adequate laboratory resources................................................................. 53
7.4 Transfer and educational pathways................................................................. 53
7.4.1 Four-year transfers................................................................. 53
7.4.2 Technical institute transfers................................................................. 54
7.4.3 Community college transfers................................................................. 54
Appendix A: Computer Engineering Body of Knowledge ............................................................... 55
A.1 Introduction.................................................................................................................................. 55
A.2 Structure of the Body of Knowledge.......................................................................................... 55
A.2.1 Core and supplementary components .................................................................................. 55
A.2.2 Assessing the time required to cover a unit ........................................................................ 56
A.2.3 Tags for KAs and KUs........................................................................................................... 56
A.3 Learning Outcomes .................................................................................................................... 56
A.4 Summary of the CE body of knowledge .................................................................................. 56
A.5 Knowledge Areas and Knowledge Units .................................................................................. 59
CE-CAE Circuits and Electronics ..................................................................................................... 59
CE-CAL Computing Algorithms ...................................................................................................... 63
CE-CAO Computer Architecture and Organization ...................................................................... 65
CE-DIG Digital Design .................................................................................................................... 68
CE-ESY Embedded Systems ............................................................................................................ 71
CE-NWK Computer Networks ...................................................................................................... 74
CE-PFP Professional Practice ........................................................................................................ 77
CE-SEC Information Security ....................................................................................................... 80
CE-SET Strategies for Emerging Technologies .......................................................................... 83
CE-SGP Signal Processing .............................................................................................................. 85
CE-SPE Systems and Project Engineering ................................................................................... 88
CE-SRM System Resource Management ..................................................................................... 92
CE-SWD Software Design ............................................................................................................. 94
Appendix B: Computer Engineering Sample Curricula ................................................................ 97
B.1 Format and Conventions .......................................................................................................... 97
B.1.1 Course Hour Conventions ..................................................................................................... 97
B.1.2 Mapping of the computer engineering BOK to a sample curriculum ................................. 98
B.1.3 Course descriptions .............................................................................................................. 98
B.2 Preparation to Enter the Profession .......................................................................................... 98
B.3 Curricula Commonalities ......................................................................................................... 99
B.4 Curriculum A: Administered by Electrical and Computer Engineering .............................. 100
B.4.1 Program Goals and Features ............................................................................................... 100
B.4.2 Summary of Requirements .................................................................................................. 100
B.4.3 Four-Year Model for Curriculum A .................................................................................... 101
B.4.4 Mapping of Computer Engineering BOK to Curriculum A ............................................. 102
B.4.5 Curriculum A – Course Summaries .................................................................................... 103
B.5 Curriculum B: Administered by Computer Science ............................................................. 106
B.5.1 Program Goals and Features ............................................................................................... 106
B.5.2 Summary of Requirements .................................................................................................. 106
B.5.3 Four-Year Model for Curriculum B .................................................................................... 107
B.5.4 Mapping of Computer Engineering BOK to Curriculum B ............................................. 108
B.5.5 Curriculum B – Course Summaries .................................................................................... 109
B.6 Curriculum C: Administered jointly by CS and EE .......................................................... 111
B.6.1 Program Goals and Features ............................................................................................... 111
B.6.2 Summary of Requirements .................................................................................................. 111
B.6.3 Four-Year Model for Curriculum C .................................................................................... 112
B.6.4 Mapping of Computer Engineering BOK to Curriculum C ............................................. 113
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], information systems [ACM/AIS, 2010], information technology [ACM/IEEECS, 2008], and software engineering [ACM/IEEECS, 2015]. Recommendations for a new information technology recommendation should appear 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 such 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, 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 it explains how the field evolved. It describes the expectations of graduates of the discipline and shows how those graduates differ from other computing disciplines. It describes the expected background, knowledge, and skills employers expect to see from graduates of computer engineering programs. These include the ability to design computer systems, the realization of the importance of practicing as professionals, and having the breadth and depth of knowledge expected of a practicing engineer. It also discusses 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 could develop or modify a curriculum to fit its needs. This body of knowledge, also known as BoK, contains broad knowledge areas that are applicable to all computer engineering programs worldwide. Each knowledge area comprises a thematic scope and a set of knowledge units. A set of learning outcomes complements each knowledge unit. The report identifies some knowledge units “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 aforementioned 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, teamwork, and a variety of laboratory experiences are fundamental to the study of computer engineering.
Additionally, the report includes sample curricula models that illustrate a methodology an institution might use to develop a curriculum in computer engineering based on its locale, its mission, and its particular 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
(Some) Contributing Reviewers

The CE2016 Steering Committee thanks the following individuals for their comments and suggestions in the development of this report.

<table>
<thead>
<tr>
<th>NAME</th>
<th>AFFILIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Olga I. Bogoiaivlenskaia</td>
<td>Petrozavodsk State University, Karelia, Russia</td>
</tr>
<tr>
<td>Iurii A. Bogoiaivlenskii</td>
<td>Petrozavodsk State University, Karelia, Russia</td>
</tr>
<tr>
<td>Qin Leihua</td>
<td>Huazhong University of Science and Technology, Wuhan, China</td>
</tr>
<tr>
<td>Zhang Liang</td>
<td>Fudan University, Shanghai, China</td>
</tr>
<tr>
<td>Clive Maynard</td>
<td>Curtin University, Perth, Australia</td>
</tr>
<tr>
<td>Doug Myers</td>
<td>Curtin University, Perth, Australia</td>
</tr>
<tr>
<td>Tang Yuhua</td>
<td>National University of Defense Technology, Changsha, China</td>
</tr>
<tr>
<td>Wang Zhiying</td>
<td>National University of Defense Technology, Changsha, China</td>
</tr>
</tbody>
</table>
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 that is ongoing even today. 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]. This report, which is an update of the CE2004 report, focuses specifically on computer engineering, referred to as CE2016.

A goal of the CE2016 effort is to revise CE2004 so that it incorporates the developments of the past decade. 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. Hence, this report contains an important ingredient that reflects 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 disciplines 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).

As the individual reports unfold to completion, representatives from the five computing disciplines have 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 appears in Figure 1.1 as taken from the overview report.

![Figure 1.1: Computing curricula reports](image)

Future curricula reports are currently under discussion. Two are these involve early discussions in 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 particular disciplines. However, they did
request that groups address a certain minimal number of matters and, consequently, that they should 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 an effort 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 aforementioned 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; its members appear 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; these contributing reviewers also appear at the beginning of this report.

### 1.3 Underlying Principles

Computer engineering is a growing and important area of endeavor. The CE2016 steering committee has 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 broadly based discipline. The final report must seek to 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 contain a broad base. To be successful, the process of creating the computer engineering recommendations must include participation from many different constituencies including industry, government, and the full range of higher educational institutions involved in computer engineering education.

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 makes every effort to ensure that the curriculum recommendations are sensitive to national and cultural differences so that they will be widely applicable throughout the world.

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 topics and concepts. Hence, topics are redundant and are not 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 the “whole 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 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. They also articulate learning outcomes, the differences between core and elective knowledge units, the number of core hours in the program, the importance of design and laboratory experiences, and various skills an individual needs to become an effective computer engineer. 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 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 two sets of references: those made within this report and a full set of references related to all computing curricula programs.

The bulk of the material in the report appears in two appendices. Appendix A addresses the body of knowledge in detail for undergraduate computer engineering programs. It includes all the computing knowledge areas, their associated knowledge units and related 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.
Chapter 2: Computer Engineering as a Discipline

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

2.1 Background

Computer engineering is 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, although intimately related, discipline. Computer engineering is solidly grounded in the theories and principles of computing, mathematics, science, and engineering and it applies these theories and principles to solve technical problems through the design of computing hardware, software, networks, and processes.

Historically, the field of computer engineering has been widely viewed as “designing computers.” In reality, the design of computers themselves has been the province of relatively few highly skilled engineers whose goal was to push forward the limits of computer and microelectronics technology. The successful miniaturization of silicon devices and their increased reliability as system building blocks and complete systems on chip has 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, multi-media and location-aware devices, wireless networks, and similar products. It also reveals itself in the myriad of applications involving embedded systems, namely those computing systems that appear in applications such as automobiles, 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. It also includes distributed computing environments (local and wide area networks, wireless networks, internets, intranets), and embedded computer systems (such as aircraft, spacecraft, and automobile control systems in which computers are embedded to perform various functions). A wide array of complex technological systems, such as power generation and distribution systems and modern processing and manufacturing plants, rely on computer systems developed and designed by computer engineers.

Technological advances and innovation continue to drive computer engineering. There is now a convergence of several established technologies (such as multimedia, computer, and networking technologies) resulting in widespread and ready access to information on an enormous scale. This has created many opportunities and challenges for computer engineers. This convergence of technologies and the associated innovation lie at the heart of economic development and the future of many organizations. The situation bodes well for a successful career in computer engineering.

2.2 Evolution of the field

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

In China, computing education in universities has developed over sixty years. The first fast developing stage was 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. They were similar to 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 of 1990s, Chinese computer education entered the third fast developing stage, with over five hundred disciplines set up in the universities. On this stage, the Ministry of Education specified ‘computer science and technology’ as the first class discipline that included ‘computer software and theory’, ‘computer systems organization’ and ‘computer applied technology’ as the second class disciplines. The knowledge of computer engineering occurred in the first class discipline, ‘computer science and technology’. After 2010, the Ministry of Education added ‘software engineering’ and ‘cyberspace security’ as new first class 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 they discontinued it. Classes or modules addressing topics in computer engineering did exist at a number of universities prior to that time. Nowadays degree courses in “computer systems engineering” or “computing and electronic systems” are far more common and they reflect an attention to wider engineering topics 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 reflecting 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 it maintains a list of accredited degree programs. Its search facility reveals that currently (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 provisions.

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 2015, the EAC of ABET has accredited over 279 computer engineering or similarly named programs, including 31 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.

<table>
<thead>
<tr>
<th>Program Name</th>
<th>Year of Initial Accreditation</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Engineering</td>
<td>10 40</td>
<td>56</td>
</tr>
<tr>
<td>Computer Systems Engineering</td>
<td>2 1</td>
<td>0</td>
</tr>
<tr>
<td>Electrical and Computer Engineering</td>
<td>9 3</td>
<td>2</td>
</tr>
<tr>
<td>(includes programs previously named EE)</td>
<td>2 5</td>
<td>2</td>
</tr>
<tr>
<td>Computer Science and Engineering</td>
<td>0 1</td>
<td>1</td>
</tr>
<tr>
<td>Other titles</td>
<td>23</td>
<td>50</td>
</tr>
<tr>
<td>Totals</td>
<td>279 (31)</td>
<td></td>
</tr>
</tbody>
</table>
One would expect that the growth trend in computer engineering will increase as computing and electronic technologies become more complex. The evolution may take many forms, including:

- 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, or
- A re-integration with electrical engineering, as computer-based systems becomes dominant in areas such as control systems and telecommunications.

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 topics in computer science and electrical engineering as well as the foundations in mathematics and sciences. Because of the rapid pace of change in the computing field, computer engineers must be life-long learners to maintain their knowledge and skills within their chosen discipline.

2.3.1 Distinctions

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

1. Possess the ability to design computers, computer-based systems and networks that include both hardware and software and 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.
2. Have a breadth of knowledge in mathematics and engineering sciences, associated with the broader scope of engineering and beyond that narrowly required for the field.
3. Acquire and maintain a preparation for professional practice in engineering.

Electrical engineering spans a wide range of areas, including bioengineering, power engineering, electronics, telecommunications and digital systems. Related to the field of computer engineering, electrical engineers concern themselves primarily with the physical aspects of electronics including circuits, signal analysis, and microelectronic devices. Computer scientists concern themselves primarily with the theoretical and algorithmic aspects of computing with a focus on the theoretical underpinnings of computing. Software engineers have a focus on the principles underlying the development and maintenance of correct (often large-scale) software throughout its lifecycle. Information systems specialists encompass the acquisition, deployment, and management of information resources for use in organizational processes. Information technology specialists would focus on meeting the needs of users within an organizational and societal context through the selection, creation, application, integration, and administration of computing technologies. Computer engineering technologists support engineers by installing and operating computer-based products, and maintaining those products.

2.3.2 Professionalism

The public has entrusted in engineers a level of responsibility because the systems they design (whether x-ray machines, air traffic control systems, or nuclear power plants) affect the public directly and indirectly. Therefore, it is incumbent upon computer engineers to exercise the utmost conscientiousness in their designs and implementations of computing systems. As such, graduates should have an understanding of the responsibilities associated with engineering practice, including the professional, societal, and ethical context in which they do their work. Such responsibilities often involve complicated trade-offs involving fiscal and social contexts. This social context encompasses a range of legal and economic issues such as intellectual property rights, security and privacy issues, liability, technological access, and global implications and uses of technologies.
Professionalism and ethics are critical elements, since the focus of engineering on design and development makes social context paramount to studies in the field. Computer engineering students must learn to integrate theory, professional practice, and social constructs in their engineering careers. It is incumbent upon all computer engineers to uphold the tenets of their profession and to adhere to the codes of professional practice. The public expects engineers to follow prescribed rules of professional practice and to not engage in activities that would tarnish their image or that of their practicing colleagues. Because of the importance of this topic, Chapter 5 of this report is devoted to an expanded discussion on professional practice and responsibilities.

2.3.3 Ability to design

Engineering draws heavily on the ability to design. The International Technology 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. Of course, novel ideas 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 topics typically include basic sciences, mathematics for both discrete and continuous domains, and applications of probability and statistics.

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

Despite differences in emphasis and content, there are certain common elements that one should expect of any computer engineering program. The Body of Knowledge, described in Chapter 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 topics. From a higher-level perspective, however, one can reasonably expect several characteristics of all computer engineering graduates. These include:

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

### 2.4 Organizational considerations

The administration of computer engineering programs falls within a variety of organizational structures. Currently, computer engineering programs are rarely organized as separate academic departments. They often appear in colleges or schools of engineering, either within an electrical engineering department, within 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 have an understanding of their responsibilities associated with engineering practice, as well as an ability to apply these principles to specific situations. Professionalism should be a constant theme that pervades the entire curriculum. In particular, the social context of engineering should be integrated into the teaching of engineering design, including the use of best practices and trade-offs among technical, fiscal, and social requirements.
In addition to professionalism, appropriate preparation encompasses both technical (design ability, laboratory experiences, use of engineering tools) and non-technical (teamwork, communication) elements. Chapter 6 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 has organized three evaluations in 2004, 2008 and 2012, respectively. The evaluation provides 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. 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) could 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, 2015] 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 should 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 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 future 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 preparing a body of knowledge commensurate with producing competent computer engineering graduates.

The CE2004 report established a collection of knowledge areas (KAs) with associated core and elective knowledge units. Within each knowledge unit were a set of topics and learning outcomes. This model has held well for the past decade and more, but it is time to revisit it. Technology has changed over this time, so a model curriculum must not only adapt with change, but also prepare graduates for technologies that are still in the inventive stage.

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, which represents a particular disciplinary subfield, not a course. 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. Compared to the CE2004 report, the CE2016 steering committee decided not to provide a list of “topics” for each knowledge area; knowledge units are comprised solely of learning outcomes. Knowledge areas also contain a “area scope” that describes the context of the particular knowledge area.

3.1.1 Core and supplementary components

One of our goals in updating the CE2004 report 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 to the curriculum compared to those that are supplementary or extra units. Core components are skills that anyone obtaining a four-year degree in the field must acquire. Supplementary components are skills that reflect expectations for advanced work according 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 all students in all degree in computer engineering programs should attain. Several learning outcomes that are important in the education of many students are not included as core and appeared as supplementary. Absence of some topics among the core components does not imply a negative judgment about their value, importance, or the relevance of those topics. Rather, it simply means that the topic is not a requirement of every student in all CE degree programs.
• The core components do not constitute a complete curriculum. It is only minimal for a complete four-year curriculum.
• It is not the case that a program should achieve core knowledge units 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.

3.1.2 Assessing the time required to cover a unit

To give readers a sense of the time required to cover a particular 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 on a topic 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 that can be at least as effective, particularly given recent improvements in educational technology. These include forms such as massive open online courses (MOOCs), 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 than 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.

### 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 they do not contain any recommended core hours.

### 3.2 Learning Outcomes

To capture the sense of what students should learn in connection with each knowledge unit, this report uses learning outcomes to describe each knowledge unit. The emphasis on learning is important. Taxonomies on 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 to do or to 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 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 outcome attainment can lead to unique expression of a range of skills in well-conceived assignments.

### 3.3 Summary of the CE body of knowledge

Table 3.1 lists the thirteen knowledge areas that form the CE body of knowledge. Table 3.2 shows the thirteen CE knowledge areas as presented in this report together with their associate 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, 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 “tens” 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

<table>
<thead>
<tr>
<th>Knowledge Area</th>
<th>CE Core Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE-CAE Circuits and Electronics</td>
<td>50 core hours</td>
</tr>
<tr>
<td>CE-ESY Embedded Systems</td>
<td></td>
</tr>
<tr>
<td>CE-DIG Digital Design</td>
<td>50 core hours</td>
</tr>
<tr>
<td>CE-CAO Computer Architecture and Organization</td>
<td>60 core hours</td>
</tr>
<tr>
<td>CE-NWK Computer Networks</td>
<td></td>
</tr>
<tr>
<td>CE-PFP Professional Practice</td>
<td></td>
</tr>
<tr>
<td>CE-SEC Information Security</td>
<td></td>
</tr>
<tr>
<td>CE-SET Strategies for Emerging Technologies</td>
<td></td>
</tr>
<tr>
<td>CE-SGP Signal Processing</td>
<td></td>
</tr>
<tr>
<td>CE-SPE Systems and Project Engineering</td>
<td></td>
</tr>
<tr>
<td>CE-SRM Systems Resource Management</td>
<td></td>
</tr>
<tr>
<td>CE-SWD Software Design</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Knowledge Areas and Knowledge Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE-CAE Circuits and Electronics</td>
</tr>
<tr>
<td>CE-CAE-1 History and overview</td>
</tr>
<tr>
<td>CE-CAE-2 Relevant tools, standards, and/or engineering constraints</td>
</tr>
<tr>
<td>CE-CAE-3 Electrical quantities and basic elements</td>
</tr>
<tr>
<td>CE-CAE-4 Electrical circuits</td>
</tr>
<tr>
<td>CE-CAE-5 Electronic materials, diodes, and bipolar transistors</td>
</tr>
<tr>
<td>CE-CAE-6 MOS transistor circuits, timing, and power</td>
</tr>
<tr>
<td>CE-CAE-7 Storage cell architecture</td>
</tr>
<tr>
<td>CE-CAE-8 Interfacing logic families</td>
</tr>
<tr>
<td>CE-CAE-9 Operational amplifiers</td>
</tr>
<tr>
<td>CE-CAE-10 Mixed-signal circuit design</td>
</tr>
<tr>
<td>CE-CAE-11 Design parameters and issues</td>
</tr>
<tr>
<td>CE-CAE-12 Circuit modeling and simulation methods</td>
</tr>
<tr>
<td>CE-CAO Computer Architecture and Organization</td>
</tr>
<tr>
<td>CE-CAO-1 History and overview</td>
</tr>
<tr>
<td>CE-CAO-2 Relevant tools, standards and/or engineering constraints</td>
</tr>
<tr>
<td>CE-CAO-3 Instruction set architecture</td>
</tr>
<tr>
<td>CE-CAO-4 Measuring performance</td>
</tr>
<tr>
<td>CE-CAO-5 Computer arithmetic</td>
</tr>
<tr>
<td>CE-CAO-6 Processor organization</td>
</tr>
<tr>
<td>CE-CAO-7 Memory system organization and architectures</td>
</tr>
<tr>
<td>CE-CAO-8 Input/Output interfacing and communication</td>
</tr>
<tr>
<td>CE-CAO-9 Peripheral subsystems</td>
</tr>
<tr>
<td>CE-CAO-10 Multi/Many-core architectures</td>
</tr>
<tr>
<td>CE-CAO-11 Distributed system architectures</td>
</tr>
<tr>
<td>CE-CAL Computing Algorithms</td>
</tr>
<tr>
<td>CE-CAL-1 History and overview</td>
</tr>
<tr>
<td>CE-CAL-2 Relevant tools, standards and/or engineering constraints</td>
</tr>
<tr>
<td>CE-CAL-3 Basic algorithmic analysis</td>
</tr>
<tr>
<td>CE-CAL-4 Algorithmic strategies</td>
</tr>
<tr>
<td>CE-CAL-5 Classic algorithms for common tasks</td>
</tr>
<tr>
<td>CE-CAL-6 Analysis and design of application-specific algorithms</td>
</tr>
<tr>
<td>CE-CAL-7 Parallel algorithms and multi-threading</td>
</tr>
<tr>
<td>CE-CAL-8 Algorithmic complexity</td>
</tr>
<tr>
<td>CE-CAL-9 Scheduling algorithms</td>
</tr>
<tr>
<td>CE-CAL-10 Basic computability theory</td>
</tr>
<tr>
<td>CE-DIG Digital Design</td>
</tr>
<tr>
<td>CE-DIG-1 History and overview</td>
</tr>
<tr>
<td>CE-DIG-2 Relevant tools, standards, and/or engineering constraints</td>
</tr>
<tr>
<td>CE-DIG-3 Number systems and data encoding</td>
</tr>
<tr>
<td>CE-DIG-4 Boolean algebra applications</td>
</tr>
<tr>
<td>CE-DIG-5 Basic logic circuits</td>
</tr>
<tr>
<td>CE-DIG-6 Modular design of combinational circuits</td>
</tr>
<tr>
<td>CE-DIG-7 Modular design of sequential circuits</td>
</tr>
<tr>
<td>CE-DIG-8 Control and datapath design</td>
</tr>
<tr>
<td>CE-DIG-9 Design with programmable logic</td>
</tr>
<tr>
<td>CE-DIG-10 System design constraints</td>
</tr>
<tr>
<td>CE-DIG-11 Fault models, testing, and design for testability</td>
</tr>
<tr>
<td>Knowledge Areas and Knowledge Units</td>
</tr>
<tr>
<td>-------------------------------------</td>
</tr>
<tr>
<td>CE-SWD</td>
</tr>
<tr>
<td>CE-SWD-1</td>
</tr>
<tr>
<td>CE-SWD-2</td>
</tr>
<tr>
<td>CE-SWD-3</td>
</tr>
<tr>
<td>CE-SWD-4</td>
</tr>
<tr>
<td>CE-SWD-5</td>
</tr>
<tr>
<td>CE-SWD-6</td>
</tr>
<tr>
<td>CE-SWD-7</td>
</tr>
<tr>
<td>CE-SWD-8</td>
</tr>
<tr>
<td>CE-SWD-9</td>
</tr>
<tr>
<td>CE-SWD-10</td>
</tr>
<tr>
<td>CE-SWD-11</td>
</tr>
<tr>
<td>CE-SWD-12</td>
</tr>
<tr>
<td>CE-SWD-13</td>
</tr>
<tr>
<td>CE-SWD-14</td>
</tr>
<tr>
<td>CE-SPE</td>
</tr>
<tr>
<td>CE-SPE-1</td>
</tr>
<tr>
<td>CE-SPE-2</td>
</tr>
<tr>
<td>CE-SPE-4</td>
</tr>
<tr>
<td>CE-SPE-5</td>
</tr>
<tr>
<td>CE-SPE-6</td>
</tr>
<tr>
<td>CE-SPE-8</td>
</tr>
<tr>
<td>CE-SPE-9</td>
</tr>
<tr>
<td>CE-SPE-10</td>
</tr>
<tr>
<td>CE-SPE-11</td>
</tr>
<tr>
<td>CE-SPE-12</td>
</tr>
<tr>
<td>CE-SET</td>
</tr>
<tr>
<td>CE-SET-1</td>
</tr>
<tr>
<td>CE-SET-2</td>
</tr>
<tr>
<td>CE-SET-3</td>
</tr>
<tr>
<td>CE-SET-4</td>
</tr>
<tr>
<td>CE-SET-5</td>
</tr>
<tr>
<td>CE-SET-6</td>
</tr>
<tr>
<td>CE-SEC</td>
</tr>
<tr>
<td>CE-SEC-1</td>
</tr>
<tr>
<td>CE-SEC-2</td>
</tr>
<tr>
<td>CE-SEC-3</td>
</tr>
<tr>
<td>CE-SEC-4</td>
</tr>
<tr>
<td>CE-SEC-5</td>
</tr>
<tr>
<td>CE-SEC-6</td>
</tr>
<tr>
<td>CE-SEC-7</td>
</tr>
<tr>
<td>CE-SEC-8</td>
</tr>
<tr>
<td>CE-SEC-9</td>
</tr>
<tr>
<td>CE-SEC-10</td>
</tr>
<tr>
<td>CE-SEC-11</td>
</tr>
<tr>
<td>CE-SRM</td>
</tr>
<tr>
<td>CE-SRM-1</td>
</tr>
<tr>
<td>CE-SRM-2</td>
</tr>
<tr>
<td>CE-SRM-3</td>
</tr>
<tr>
<td>CE-SRM-4</td>
</tr>
<tr>
<td>CE-SRM-5</td>
</tr>
<tr>
<td>CE-SRM-6</td>
</tr>
<tr>
<td>CE-SRM-7</td>
</tr>
<tr>
<td>CE-SRM-8</td>
</tr>
<tr>
<td>CE-SGP</td>
</tr>
<tr>
<td>CE-SGP-1</td>
</tr>
<tr>
<td>CE-SGP-2</td>
</tr>
<tr>
<td>CE-SGP-3</td>
</tr>
<tr>
<td>CE-SGP-4</td>
</tr>
<tr>
<td>CE-SGP-5</td>
</tr>
<tr>
<td>CE-SGP-7</td>
</tr>
<tr>
<td>CE-SGP-8</td>
</tr>
<tr>
<td>CE-SGP-9</td>
</tr>
<tr>
<td>CE-SGP-10</td>
</tr>
<tr>
<td>CE-NWK</td>
</tr>
<tr>
<td>CE-NWK-1</td>
</tr>
<tr>
<td>CE-NWK-2</td>
</tr>
<tr>
<td>CE-NWK-3</td>
</tr>
<tr>
<td>CE-NWK-4</td>
</tr>
<tr>
<td>CE-NWK-5</td>
</tr>
<tr>
<td>CE-NWK-6</td>
</tr>
<tr>
<td>CE-NWK-7</td>
</tr>
<tr>
<td>CE-NWK-8</td>
</tr>
<tr>
<td>CE-NWK-9</td>
</tr>
<tr>
<td>CE-NWK-10</td>
</tr>
<tr>
<td>CE-NWK-11</td>
</tr>
<tr>
<td>CE-ESY</td>
</tr>
<tr>
<td>CE-ESY-1</td>
</tr>
<tr>
<td>CE-ESY-2</td>
</tr>
<tr>
<td>CE-ESY-3</td>
</tr>
<tr>
<td>CE-ESY-4</td>
</tr>
<tr>
<td>CE-ESY-5</td>
</tr>
<tr>
<td>CE-ESY-6</td>
</tr>
<tr>
<td>CE-ESY-7</td>
</tr>
<tr>
<td>CE-ESY-8</td>
</tr>
<tr>
<td>CE-ESY-9</td>
</tr>
<tr>
<td>CE-ESY-10</td>
</tr>
<tr>
<td>CE-ESY-11</td>
</tr>
<tr>
<td>CE-ESY-12</td>
</tr>
<tr>
<td>CE-ESY-13</td>
</tr>
<tr>
<td>CE-SPE</td>
</tr>
<tr>
<td>CE-SPE-1</td>
</tr>
<tr>
<td>CE-SPE-2</td>
</tr>
<tr>
<td>CE-SPE-3</td>
</tr>
<tr>
<td>CE-SPE-4</td>
</tr>
<tr>
<td>CE-SPE-5</td>
</tr>
<tr>
<td>CE-SPE-6</td>
</tr>
<tr>
<td>CE-SPE-7</td>
</tr>
<tr>
<td>CE-SPE-8</td>
</tr>
<tr>
<td>CE-SPE-9</td>
</tr>
<tr>
<td>CE-SPE-10</td>
</tr>
<tr>
<td>CE-SPE-11</td>
</tr>
<tr>
<td>CE-SET</td>
</tr>
<tr>
<td>CE-SET-1</td>
</tr>
<tr>
<td>CE-SET-2</td>
</tr>
<tr>
<td>CE-SET-3</td>
</tr>
<tr>
<td>CE-SET-4</td>
</tr>
<tr>
<td>CE-SET-5</td>
</tr>
<tr>
<td>CE-SET-6</td>
</tr>
<tr>
<td>CE-SRM</td>
</tr>
<tr>
<td>CE-SRM-1</td>
</tr>
<tr>
<td>CE-SRM-2</td>
</tr>
<tr>
<td>CE-SRM-3</td>
</tr>
<tr>
<td>CE-SRM-4</td>
</tr>
<tr>
<td>CE-SRM-5</td>
</tr>
<tr>
<td>CE-SRM-6</td>
</tr>
<tr>
<td>CE-SRM-7</td>
</tr>
<tr>
<td>CE-SRM-8</td>
</tr>
</tbody>
</table>


### 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 of topics 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 that emphasize what the steering committee considers essential to computer engineering. Notwithstanding these recommendations, a program should evaluate its own goals and choose the required mathematics content needed for its graduates to achieve success in industry or in graduate studies. See Chapter 6 for a more detailed discussion on mathematics in computer engineering.

<table>
<thead>
<tr>
<th>Mathematics Knowledge</th>
<th>Areas and Units</th>
</tr>
</thead>
</table>

### 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 study include as much natural science (e.g., biology or chemistry in addition to physics) as appropriate so students obtain a command of the scientific bases for engineering topics. 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.

### 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: Body of Knowledge: CE2016 vs. CE2004 Knowledge Areas

<table>
<thead>
<tr>
<th>CE2016 Knowledge Areas</th>
<th>Core Hours</th>
<th>CE2004 Knowledge Areas</th>
<th>Core Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE-CAE Circuits and Electronics</td>
<td>50</td>
<td>CE-CSG Circuits and Signals</td>
<td>43</td>
</tr>
<tr>
<td>CE-CAL Computing Algorithms</td>
<td>30</td>
<td>CE-ALG Algorithms</td>
<td>30</td>
</tr>
<tr>
<td>CE-CAO Computer Architecture &amp; Organization</td>
<td>60</td>
<td>CE-CAO Computer Architecture &amp; Organization</td>
<td>63</td>
</tr>
<tr>
<td>CE-SPE Systems and Project Engineering</td>
<td>30</td>
<td>CE-CSE Computer Systems Engineering</td>
<td>18</td>
</tr>
<tr>
<td>CE-DIG Digital Design</td>
<td>50</td>
<td>CE-DIG Digital Logic</td>
<td>57</td>
</tr>
<tr>
<td>CE-ESY Embedded Systems</td>
<td>40</td>
<td>CE-ESY Embedded Systems</td>
<td>20</td>
</tr>
<tr>
<td>CE-NWK Computer Networks</td>
<td>20</td>
<td>CE-NWK Computer Networks</td>
<td>21</td>
</tr>
<tr>
<td>CE-PFP Professional Practice</td>
<td>20</td>
<td>CE-SPR Social and Professional Issues</td>
<td>16</td>
</tr>
<tr>
<td>CE-SEC Information Security</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE-SET Strategies for Emerging Technologies</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE-SGP Signal Processing</td>
<td>30</td>
<td>CE-DSP Digital Signal Processing</td>
<td>17</td>
</tr>
<tr>
<td>CE-SRM Systems Resource Management</td>
<td>20</td>
<td>CE-OPS Operating Systems</td>
<td>20</td>
</tr>
<tr>
<td>CE-SWD Software Design</td>
<td>40</td>
<td>CE-SWE Software Engineering</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-DBS Database Systems</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-ELE Electronics</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-HCI Human-Computer Interaction</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-PRF Programming Fundamentals</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-VLS VLSI Design and Fabrication</td>
<td>10</td>
</tr>
</tbody>
</table>

Total: 420

Table 3.4 shows that the body of knowledge for CE2016 has 13 knowledge areas while the body of knowledge for CE2004 has 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 modern approach.

3.5 Rationale for 420 hours in computer engineering

As already mentioned, the CE2016 body of knowledge contains 420 core hours. Additionally, the CE2016 mathematics recommendation contains 120 core hours. The numbers “420” and “120” are not arbitrary.

Within a typical academic context, a four-year undergraduate program has at least 120 semester hours or credits. 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 15 weeks. Hence, thirty semester hours (one year of study) is equivalent to 30×15 = 450 lecture hours.

It is common to include examination time within the fifteen weeks of a semester. However, if we exclude examination time, which might be the equivalent of one week during a fifteen-week semester, then one semester hour (credit) is equivalent to fourteen lecture hours plus one examination hour. Hence, one year of study excluding examinations is equivalent to 30×14 = 420 lecture hours, or simply 420 hours. The CE2016 steering committee believes that 420 core hours reflects a minimal curricular component needed to produce a competent computer engineer.

The steering committee also believes a student in computer engineering should experience at least the equivalent of 8 semester hours (or credits) in mathematics. Using the above discussion, this generates 8×15 = 120 lecture hours. 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. Typically, three curricular models exist for undergraduate programs, which include three-year, four-year, and five-year models.

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 studies, additional engineering, and other topics. 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 studies, additional engineering, and other topics. The model is adaptable to many worldwide systems of study. A five-year model reflects recent advances in Europe due to the Bologna Accord. 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 (and, in particular, computer engineering), teaching them to think and function as engineers, and setting expectations for the future. Preparation for practice is essential because a majority of 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 expect 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, and 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 given 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 Design in the curriculum

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

4.2.1 Design throughout the curriculum

The principles of engineering design must pervade the entire computer engineering curriculum to produce competent graduates. Throughout their education, computer engineering students should encounter different approaches to design so that they become familiar with the strengths and weaknesses of these approaches. Typically, the context in which design occurs provides a framework 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 students’ accumulated technical knowledge and prior design experiences.

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

At a different level are all of 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 technologies. Similarly, one can make the same argument for issues in hardware design. In short, design is central to computer engineering.

4.2.2 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, have the opportunity to demonstrate their ability 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:

- Engagement at the level of the structure and function of the system as a whole, 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 particular knowledge and skills in many ways. This might include a demonstration, a presentation, an oral examination, production of a web site, industry review, etc. 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 topics will be covered before the culminating design experience, some may best be introduced during it.

4.3 The laboratory experience

The laboratory experience is an essential part of the computer engineering curriculum and serves multiple functions. As in any engineering curriculum, it is important that computer engineering students have many opportunities to observe, explore, and manipulate characteristics and behaviors of actual devices, systems, and processes. This includes designing, implementing, testing, and documenting hardware and software, designing experiments to acquire data, analyzing and interpreting that data, and using that data to correct or improve the design.

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

Laboratories should include some physical implementation of designs such as electronic and digital circuits, breadboarding, FPGAs, microprocessor interfacing, prototyping, and implementation of firmware. Laboratories should also include application and simulation software to design digital and computer systems. The use of simulation tools to model and study real systems is often desirable and necessary to allow students to study systems that are not practical to design and implement given the available time and resources. Such tools would also be useful where it might be difficult to acquire the detailed information necessary to study their behavior. 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 their measured/observed results whether good or bad. This also offers opportunities to record trade-offs and to explore the effects of those design trade-offs. 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
• 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, efficiency, etc.) 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. The laboratory experience capitalizes on this interest to provide a foundation for other important elements of practical activity. Fundamentally, carefully planned practical assignments in a laboratory setting should help students develop confidence in their technical ability. The laboratory experience should help students develop the expertise to build new devices and to appreciate the important role of technical staff, workshop teams, and professionals from other disciplines.

4.4 The role of engineering tools

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

For the computer engineer, the relevant range of tools spans the whole hardware and software spectrum. Hardware design and analysis tools include instruments for measuring and analyzing hardware behavior, hardware description language and other design tools, simulators and emulators, and debugging tools. Other hardware tools include those to support circuit design, printed circuit design layout, circuit behavior analysis, block diagrams 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, integrated development environments (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 all 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 all of its guises: safety, usability, reliability, security, and so on.

### 4.5 Applications of computer engineering principles

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

- Motivating students in their studies
- Guiding 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 might accomplish such work in several ways. Some approaches might include an extended internship experience or the equivalent of a full semester’s work that would count toward a major in that discipline. Some institutions offer cooperative education programs in which students alternate terms of study and engineering work in industry. Activities of this kind can be interdisciplinary in nature and provide opportunities for particularly beneficial kinds of group activity. Thus, the computer engineer might work with professionals from other disciplines, such as computer scientists, electrical engineers, mechanical engineers, entrepreneurs, financial experts, marketers, and product designers.

### 4.6 Complementary skills

In today’s world, there are pressures on institutions to ensure that graduates have the capacity to meet the needs of employers. A more positive view is that institutions can be agents of change, producing graduates who are capable of moving into employment with skills and expectations that benefit their employers.
One aspect of this is ensuring that students possess a set of transferable or personal skills such as communication skills, teamwork skills, 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, one should give students formative feedback on these skills in the context of computer engineering and in a manner that highlights their relevance and importance to the discipline.

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

4.6.1 Communication skills

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

One particular aspect of the activity of a computer engineer is to convey and negotiate project requirements and implementation details to a workshop or to 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 (for example, specifications, requirements, safety cases, documentation) as well as report writing, which should address 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 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.

4.6.2 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. Many times 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 extend over a reasonably long period of time, possibly a full semester or even longer.

Many of the problems of teamwork relate to communication skills. Where multidisciplinary 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 have the opportunity to acquire these skills as undergraduates, such as

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

4.6.3 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]. Continuing, “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.

4.6.4 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 would 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 their focus is on the practicum and not on passing exams. Any constructive experience a student can acquire 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 expects 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.

4.6.5 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.

A number of basic strategies are appropriate. 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 by 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.

4.6.6 Business perspectives

To complement the technical side of their experiences, computer engineers need to have 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 multidisciplinary in nature

In addition, students need to 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 topics might 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.

4.7 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 nonoverridable smart systems to wearable computing [ALIBAB].

In addition to focusing on the industry and gaining valuable work experience while attending a university, it is important that students nearing graduation are ready for important interviews by 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 in the course of 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.

4.8 The elements of an engineering education

In summary, proper preparation for professional practice should result in graduates who are capable of the following:
• Seeing their discipline as based on sound principles and sound underpinnings, to recognize what these are, and to be able to apply them
• Understanding the important relationship between theory and practice
• Placing importance on design and being able to select appropriate approaches in particular contexts
• Recognizing the importance of understanding 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 teamwork 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.
Chapter 5: Professional Practice

As noted earlier 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 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, which 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 ways 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. As a result, 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 particular importance are issues related to intellectual property. An understanding of patent law is important, particularly when the companies for whom they work may have an active patent program or when entrepreneurs develop their own products. It is also necessary to understand copyrights 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.

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

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

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

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

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. CE programs could foster these skills in computing courses. Alternatively, programs in 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 may be developed solely for the class, but 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 jurisdiction of different country courts. Topics included may be history of computing, impact of computers on society, computing careers, legal and ethical responsibilities, international Computer Laws and the computing profession.

- **Practicum/Internship/Co-op Programs:** These programs are sponsored by the institution (preferably) or department to allow students to have 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 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 in some cases may also receive course credit.

- **Team-based Implementation Courses:** These courses emphasize the process of CE system development and typically include a team project. Course topics include development processes, project management, economics, risk management, requirements engineering, design, implementation, maintenance, software and hardware retirement, system quality assurance, ethics, and teamwork. Topic 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, which 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 practice. 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. As students progress into their second year courses, they can start to keep records of their work as a professional might do 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. As a result, 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 topics appears under the professional practice (CE-PFP) 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 well alumni 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 make sure 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. In their role as the primary consumer of graduating students, industry and government 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 institutions that offer them, in that a student who has an internship with a company may choose to work there again 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 a number of ways:

- They can function in the role of mentors to students working on projects.
- They can give special presentations to classes telling students and faculty about their firm, their work, and their development processes.
- They can take part-time positions as adjunct instructors to strengthen a university's course offerings.
• They can provide in-house training materials and/or classes to faculty and students in specialized research, process, or software tool areas.
• They can serve on industrial advisory boards, which service allows them to provide valuable feedback to the department and institution about the strengths and weaknecesses 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 particular 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/awards as well as faculty teaching/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/public sectors help to ensure the viability/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/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. As a result, it is not advisable to focus attention in the curriculum on a particular 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 immaturly 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 later in Chapter 6, learning how to work in teams is not a natural process for many students, but it is nonetheless 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. We can base assessment on 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 project 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 all 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

These codes also provide public information concerning the precepts considered central to the profession. These codes provide a level playing field for professionals with the prospects of avoiding ethical dilemmas whenever possible and helping professionals “do the right thing” when faced with ethical decision making during their course of professional practice. In computing, these codes are often binding upon the members of a society and they provide guidance in helping professionals make decisions affecting their practice.

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]
- International Federation for Information Processing (IFIP): *Harmonization of Professional Standards and also Ethics of Computing* [IFIP, 1998]
- Association of Information Technology Professionals (AITP): *AITP Code of Ethics and the AITP Standards of Conduct* [AITP, 2002]

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

### 5.5.2 Education and professional practice

CE 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 particular 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 issues of 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 (an entire program of study) is far from straightforward. The body of knowledge introduced in Chapter 4, and presented in Appendix A, provides a starting point, but many other influences contribute to the creation of the curriculum. The purpose of this chapter is to explore issues in the design and creation of a complete computer engineering degree program. These issues include specifics such as packaging material from the BOK into courses, determining required mathematics and science courses, and more general they include considerations such as creating an overall style or ethos for a particular 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 particular approaches can be confident that they will continue to enjoy and be successful at the more advanced levels.

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

This view requires tempering by consideration of the students’ point of view. Students who choose to study computer engineering are often motivated by the hands-on nature of engineering, as well as their prior experience with computers. During their initial academic terms, if students only take courses on mathematics and science, without obvious computer engineering applications, it may create a situation of frustration and disillusionment. It is desirable to position topics involving very new topics in the later years. These new topics are often at the forefront of research and development and after studying them, students can genuinely claim to be up-to-date in their subject area. That is important since they enter industry or employment as the agents of technology change and transfer. Other considerations will also influence the characteristics of a particular degree program. These considerations include:

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

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

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

- There may be enough students in computer engineering to justify the provision of specialist courses devised...
solely for computer engineering students

- Alternatively, computer engineers might attend classes offered from the computer science and electrical engineering curricula with additional selected classes being mounted specifically to address the specialist topics for computer engineering students
- Additional possibilities also exist depending on local arrangements and circumstances

6.2 Principles for Curriculum Design

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

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

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

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

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

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 different 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 knowledge areas. However, coverage of a BOK knowledge area can also be done by spreading the knowledge units among several courses. The Computer Science Curricula 2013 in fact recommends that information security topic coverage be integrated through the curriculum. Knowledge units from the Professional Practice and Strategies for Emerging Technologies knowledge areas could similarly be integrated into several courses across a curriculum. It is recognized that spreading knowledge unit coverage among several courses does take more coordination and planning among the faculty members responsible for those courses.
6.3.2 Course Arrangement

Course sequences are typically used for topic coverage, with the sequence following introductory, intermediate, and advanced topics 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 to these topics and expose students to state of the art techniques in these areas. Intermediate courses also reinforce topics from introductory courses, such as 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 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, choices for technical elective courses should not be overly restrictive, such as allowing any upper level course from computer science or electrical and computer engineering to serve as a technical elective.

6.3.3 Lab Experiences

Lab experiences are crucial to developing skills required to practice the art of computer engineering. Good lab experiences force a student to demonstrate they understand lecture topics by successfully creating a program or hardware system to meet a set of criteria. Good lab experiences also expose the student 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 a significant faculty time and departmental financial support, but emphasizes that these investments are needed in order to produce quality computer engineering graduates. Sometimes innovative approaches are needed for adding lab experiences, perhaps by taking a traditional 3 hour semester lecture course and transforming it into a 2 lecture + 1 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

Chapter 5 has already described several methods for incorporating elements of professional practice into a computer engineering curriculum. Professional practice topic coverage is important to developing well-rounded computer engineering graduates.

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 achieving these skills, it is essential that appropriate communication requirements be included within computer engineering courses. This is necessary to ensure that students have the ability to communicate discipline-specific content; further, such activities contribute to the students’ learning of technical material.
6.4 Course Material Presented by Other Departments

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

6.4.1 Mathematics Requirements

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

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

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

The CE2004 report contains the 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, Linear Algebra). One should note that the majority of computer engineering programs already included coverage of these knowledge areas, and this is simply an acknowledgement of that fact. The reasoning for including these knowledge areas are as follows:

- **Discrete structures**: All students need knowledge of the mathematical principles of discrete structures and exposure to related tools. All programs should include enough exposure to this area to cover the core topics specified in the computer engineering body of knowledge.
- **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 and it is fundamental to all engineering programs.
- **Probability and statistics**: These related topics underpin considerations of reliability, safety, dependence, and various other concepts of concern to the computer engineer. Many programs will have students take an existing course in probability and statistics; some programs may allow some students to study less than a full semester course in the subject. Regardless of the implementation, all students should get at least some brief exposure to discrete and continuous probability, stochastic processes, sampling distributions, estimation, hypothesis testing, and correlation and regression, as specified in the computer engineering body of knowledge.
- **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 to develop their sophistication in this area and to support classes
in topics such as communications theory, security, signals and systems, analog electronics, and artificial intelligence. That mathematics might consist of courses in any number of areas, including further calculus, 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 hand-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 experience this mode of inquiry in courses that provide some exposure to laboratory work, including data collection and analysis.
- Students may acquire their scientific perspective in any of a variety of domains, depending on program objectives and their area of interest.

6.4.3 Other Requirements

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

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 remaining examples reflect possible curricula reflecting those 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 topics included in 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. As a result, 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 according to the specific interests and knowledge base of the department, particularly at smaller institutions where expertise is concentrated in particular areas.

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

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, makes it difficult to attract and retain faculty. Institutions will need to have an aggressive plan to recruit and retain faculty. Incentives such as hiring packages and modified teaching responsibilities may prove advantageous for this endeavor.

While the computer engineering program may draw on faculty from related disciplines, as a professional field there must be a core faculty with appropriate professional training and experience. Additionally, faculty members must maintain currency with developments in the field. Institutions must make appropriate accommodations for the professional development of faculty, whether achieved through research, conference participation, 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 typically has many required laboratories included in the curriculum. The laboratory component leads to an increased need for staff to assist in both the development of materials and the teaching of laboratory sections. This development will add to the academic support costs of a high-quality computer engineering program. Close contacts with relevant industry can lead to the ready availability of interesting up-to-date case study material, but also can open up 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

Access, retention, and degree attainment across bachelor’s academic CE programs are shaped by multiple factors such as admission criteria, student academic preparedness, the diversity of student population, life experiences, and different configurations of educational pathways. 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 bachelor’s CE degree programs from a global perspective.

#### 7.4.1 Four-year transfers

Understanding the entry points into, and the pathways through, bachelor’s CE programs will help structure these pathways to serve students, companies, and the computing industry as a whole. In particular, appreciating variations and compatibilities in these pathways across the regions of the world will help enable a global computing workforce.
The steering committee considered pathways toward a bachelor’s degree 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 was 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 a four-year computer engineering program at a university. The contexts of the two programs would be very different. Students attending technical institutes would like not study 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 even 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, in some states students 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-PFP** Professional Practice
- **CE-SEC** Information Security
- **CE-SET** Strategies for Emerging Technologies
- **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 CE body of knowledge has a three-level hierarchical structure. The highest level of the hierarchy is the knowledge area, which represents a particular disciplinary subfield, not a course. 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. Knowledge units comprise solely of learning outcomes. Knowledge areas also contain an “area scope” that describes the context of the particular knowledge area.

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 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 to the curriculum compared to those that are supplementary or extra units. Core components are skills that anyone obtaining a four-year degree in the field should acquire. Supplementary components are skills that reflect expectations for advanced work according to the needs of a program.

The steering committee has found it helpful to emphasize the following points.
• The core components refer to the knowledge and skills students in all degree programs in computer engineering should achieve. Several learning outcomes that are important in the education of many students are not included as core and appeared as supplementary.
• The core components do not constitute a complete curriculum.
• Core knowledge units do not imply exposure within a set of introductory courses early in the four-year curriculum. While many of the knowledge and skills defined as core are indeed introductory, students can achieve other core KUs only after they have developed significant background in their studies.

A.2.2 Assessing the time required to cover a unit

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 on a topic 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. 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 than one core hour, independent of the teaching style.
• The hours specified do not include time spent outside of class such as instructor preparation time or the time students spend outside of class. Usually, the amount of out-of-class work is approximately three times the in-class time.
• The hours listed for a knowledge unit represent a minimum level of coverage. Instructors may find that delivery of material to the level of depth that they wish to incorporate will take much longer than this.

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.3 Learning Outcomes

To capture the various levels of ability associated with acquiring knowledge this report uses learning outcomes to describe each knowledge unit. The emphasis on learning is important. Taxonomies on verbs such as “define” or “evaluate” are useful to describe the expected depth of learning. 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 to do or to know by the time of graduation. In this report, the steering committee has tried to limit the number of learning outcomes to emphasize essential knowledge and skills.

A.4 Summary of the CE body of knowledge

Table A.1 lists the thirteen knowledge areas that form the CE body of knowledge together with their associate knowledge units. The table also shows the 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” is the fifth knowledge unit of the “embedded systems” knowledge area and that it 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>
<thead>
<tr>
<th>Knowledge Areas and Units</th>
<th>CE-CAL Computing Algorithms [30 core hours]</th>
</tr>
</thead>
<tbody>
<tr>
<td>History, overview, and principles</td>
<td>CE-CAL-1 History and overview [1]</td>
</tr>
<tr>
<td>Relevant tools, standards, and/or engineering constraints</td>
<td>CE-CAL-2 Relevant tools, standards and/or engineering constraints [1]</td>
</tr>
<tr>
<td>Basic algorithmic analysis</td>
<td>CE-CAL-3 Basic algorithmic analysis [4]</td>
</tr>
<tr>
<td>Algorithmic strategies</td>
<td>CE-CAL-4 Algorithmic strategies [6]</td>
</tr>
<tr>
<td>Classic algorithms for common tasks</td>
<td>CE-CAL-5 Classic algorithms for common tasks [3]</td>
</tr>
<tr>
<td>Analysis and design of application-specific algorithms</td>
<td>CE-CAL-6 Analysis and design of application-specific algorithms [6]</td>
</tr>
<tr>
<td>Parallel algorithms and multi-threading</td>
<td>CE-CAL-7 Parallel algorithms and multi-threading [6]</td>
</tr>
<tr>
<td>Algorithmic complexity</td>
<td>CE-CAL-8 Algorithmic complexity [3]</td>
</tr>
<tr>
<td>Scheduling algorithms</td>
<td>CE-CAL-9 Scheduling algorithms</td>
</tr>
<tr>
<td>Basic computability theory</td>
<td>CE-CAL-10 Basic computability theory</td>
</tr>
<tr>
<td>Instruction set architecture</td>
<td>CE-CAO-1 Instruction set architecture [10]</td>
</tr>
<tr>
<td>Processor organization</td>
<td>CE-CAO-6 Processor organization [10]</td>
</tr>
<tr>
<td>Memory system organization and architectures</td>
<td>CE-CAO-7 Memory system organization and architectures [9]</td>
</tr>
<tr>
<td>Input/Output interfacing and communication</td>
<td>CE-CAO-8 Input/Output interfacing and communication [7]</td>
</tr>
<tr>
<td>Peripheral subsystems</td>
<td>CE-CAO-9 Peripheral subsystems [7]</td>
</tr>
<tr>
<td>Multi/Many-core architectures</td>
<td>CE-CAO-10 Multi/Many-core architectures [6]</td>
</tr>
<tr>
<td>Distributed system architectures</td>
<td>CE-CAO-11 Distributed system architectures [3]</td>
</tr>
<tr>
<td>History and overview</td>
<td>CE-CAP-1 History and overview [1]</td>
</tr>
<tr>
<td>Relevant tools, standards, and/or engineering constraints</td>
<td>CE-CAP-2 Relevant tools, standards and/or engineering constraints [1]</td>
</tr>
<tr>
<td>Basic software techniques for embedded applications</td>
<td>CE-CAP-3 Basic software techniques for embedded applications [3]</td>
</tr>
<tr>
<td>Parallel input and output</td>
<td>CE-CAP-5 Parallel input and output [3]</td>
</tr>
<tr>
<td>Asynchronous and synchronous serial communication</td>
<td>CE-CAP-6 Asynchronous and synchronous serial communication [6]</td>
</tr>
<tr>
<td>Data acquisition, control, sensors, actuators</td>
<td>CE-CAP-8 Data acquisition, control, sensors, actuators [3]</td>
</tr>
<tr>
<td>Techniques for low-power operation</td>
<td>CE-CAP-10 Techniques for low-power operation [3]</td>
</tr>
<tr>
<td>Mobile and networked embedded systems</td>
<td>CE-CAP-11 Mobile and networked embedded systems [3]</td>
</tr>
<tr>
<td>Advanced Input/output topics</td>
<td>CE-CAP-12 Advanced Input/output topics [3]</td>
</tr>
<tr>
<td>Computing platforms for embedded systems</td>
<td>CE-CAP-13 Computing platforms for embedded systems</td>
</tr>
<tr>
<td>History and overview</td>
<td>CE-CAP-1 History and overview [1]</td>
</tr>
<tr>
<td>Relevant tools, standards, and/or engineering constraints</td>
<td>CE-CAP-2 Relevant tools, standards and/or engineering constraints [1]</td>
</tr>
<tr>
<td>Basic software techniques for embedded applications</td>
<td>CE-CAP-3 Basic software techniques for embedded applications [3]</td>
</tr>
<tr>
<td>Parallel input and output</td>
<td>CE-CAP-5 Parallel input and output [3]</td>
</tr>
<tr>
<td>Asynchronous and synchronous serial communication</td>
<td>CE-CAP-6 Asynchronous and synchronous serial communication [6]</td>
</tr>
<tr>
<td>Data acquisition, control, sensors, actuators</td>
<td>CE-CAP-8 Data acquisition, control, sensors, actuators [3]</td>
</tr>
<tr>
<td>Techniques for low-power operation</td>
<td>CE-CAP-10 Techniques for low-power operation [3]</td>
</tr>
<tr>
<td>Mobile and networked embedded systems</td>
<td>CE-CAP-11 Mobile and networked embedded systems [3]</td>
</tr>
<tr>
<td>Advanced Input/output topics</td>
<td>CE-CAP-12 Advanced Input/output topics [3]</td>
</tr>
<tr>
<td>Computing platforms for embedded systems</td>
<td>CE-CAP-13 Computing platforms for embedded systems</td>
</tr>
</tbody>
</table>

Table A-1: CE2016 Body of Knowledge (CE Core Hours: 420)
<table>
<thead>
<tr>
<th>Knowledge Areas and Units</th>
<th>Core Hours</th>
<th>CE-SGP</th>
<th>CE-SRM</th>
<th>CE-SWD</th>
<th>CE-SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE-SGP: Signal Processing</td>
<td>30 hours</td>
<td>CE-SGP-1</td>
<td>CE-SRM-1</td>
<td>CE-SWD-1</td>
<td>CE-SET-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SGP-2</td>
<td>CE-SRM-2</td>
<td>CE-SWD-2</td>
<td>CE-SET-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SGP-3</td>
<td>CE-SRM-3</td>
<td>CE-SWD-3</td>
<td>CE-SET-3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SGP-4</td>
<td>CE-SRM-4</td>
<td>CE-SWD-4</td>
<td>CE-SET-4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SGP-5</td>
<td>CE-SRM-5</td>
<td>CE-SWD-5</td>
<td>CE-SET-5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SGP-6</td>
<td>CE-SRM-6</td>
<td>CE-SWD-6</td>
<td>CE-SET-6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SGP-7</td>
<td>CE-SRM-7</td>
<td>CE-SWD-7</td>
<td>CE-SET-7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SGP-8</td>
<td>CE-SRM-8</td>
<td>CE-SWD-8</td>
<td>CE-SET-8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SGP-9</td>
<td></td>
<td>CE-SWD-9</td>
<td>CE-SET-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SGP-10</td>
<td></td>
<td>CE-SWD-10</td>
<td>CE-SET-10</td>
</tr>
<tr>
<td>CE-SRM: Systems Resource Management</td>
<td>20 hours</td>
<td>CE-SRM-1</td>
<td>CE-SRM-2</td>
<td>CE-SWD-1</td>
<td>CE-SET-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SRM-3</td>
<td>CE-SRM-4</td>
<td>CE-SWD-2</td>
<td>CE-SET-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SRM-5</td>
<td>CE-SRM-6</td>
<td>CE-SWD-3</td>
<td>CE-SET-3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SRM-7</td>
<td>CE-SRM-8</td>
<td>CE-SWD-4</td>
<td>CE-SET-4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SRM-9</td>
<td></td>
<td>CE-SWD-5</td>
<td>CE-SET-5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SRM-10</td>
<td></td>
<td>CE-SWD-6</td>
<td>CE-SET-6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SRM-11</td>
<td></td>
<td>CE-SWD-7</td>
<td>CE-SET-7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SRM-12</td>
<td></td>
<td>CE-SWD-8</td>
<td>CE-SET-8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SRM-13</td>
<td></td>
<td>CE-SWD-9</td>
<td>CE-SET-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SRM-14</td>
<td></td>
<td>CE-SWD-10</td>
<td>CE-SET-10</td>
</tr>
<tr>
<td>CE-SWD: Software Design</td>
<td>40 hours</td>
<td>CE-SWD-1</td>
<td>CE-SWD-2</td>
<td>CE-SWD-1</td>
<td>CE-SET-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SWD-3</td>
<td>CE-SWD-4</td>
<td>CE-SWD-2</td>
<td>CE-SET-2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SWD-5</td>
<td>CE-SWD-6</td>
<td>CE-SWD-3</td>
<td>CE-SET-3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SWD-7</td>
<td>CE-SWD-8</td>
<td>CE-SWD-4</td>
<td>CE-SET-4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SWD-9</td>
<td>CE-SWD-10</td>
<td>CE-SWD-5</td>
<td>CE-SET-5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SWD-11</td>
<td>CE-SWD-12</td>
<td>CE-SWD-6</td>
<td>CE-SET-6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SWD-13</td>
<td>CE-SWD-14</td>
<td>CE-SWD-7</td>
<td>CE-SET-7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SWD-14</td>
<td>CE-SWD-15</td>
<td>CE-SWD-8</td>
<td>CE-SET-8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SWD-15</td>
<td>CE-SWD-16</td>
<td>CE-SWD-9</td>
<td>CE-SET-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CE-SWD-16</td>
<td>CE-SWD-17</td>
<td>CE-SWD-10</td>
<td>CE-SET-10</td>
</tr>
</tbody>
</table>

**CE-SET: Strategies for Emerging Technologies**

- CE-SET-1: History and overview [3]
- CE-SET-2: Relevant tools, standards, and/or engineering constraints [2]
- CE-SET-3: Conceptual emerging technologies [2]
- CE-SET-5: Additional emerging technologies [1]
- CE-SET-6: Tradeoffs in strategies for emerging technologies

**CE-SPE: Systems and Project Engineering**

- 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: Human-computer interaction [3]
- 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-10: Concurrent hardware and software design [2]
- CE-SPE-11: System integration, testing and validation [2]
- CE-SPE-12: Maintainability, sustainability, manufacturability [2]
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.
- Use Ohm’s law to analyze basic electrical circuits.
- Articulate 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:

- Compare various elements of circuit models including independent and dependent sources, 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 utilize phasor representation of voltage and current in analyzing circuits.
- Explain the response of electrical circuits from sinusoidal signal excitation.
- Define and utilize impedance and admittance as well as source transformations.
- Use circuit simulators to model and analyze simple resistive and RLC circuits.

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: 8 hours

Core Learning Outcomes:

- Explain characteristics and properties of electronic materials to include 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.
- Contrast the difference between Schottky, Zener, and variable capacitance diodes.
- Design a single diode circuit and describe the significance of a load line.
- 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.
- Provide examples of bipolar transistors used in the construction of a range of common circuits.
- Produce mathematical models to represent material properties of electronic devices.
- Provide examples of using mathematical models in circuit simulators.

Elective Learning Outcomes:

- Illustrate multidiode circuits such as rectifiers and direct current (DC) involving DC-DC voltage level converters.
- Design a multidiode circuit to include rectifiers.
- Design a multidiode circuit to include DC-DC voltage level converters.
- Implement diode logic using only AND and OR functions.

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: (Note: see also CE-DIG-7, which focuses on design attributes of storage cells)
- Compare and contrast the circuit properties of implementations of various storage elements (e.g., latches, flip-flops, clocked registers).
- Compare and 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:
- Compare and 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 for transistor-transistor logic (TTL) and emitter-coupled logic (ECL) to include TTL/CMOS, TTL/ECL, and CMOS/ECL.
- 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 to include passive, active, DC, and alternating current (AC) features.

CE-CAE-9 Operational amplifiers
Minimum core coverage time: 2 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:
- Compare 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.
- Compare 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.
- Contrast the properties that distinguish particular kinds of D/A and A/D converters.
- Interpret and apply D/A and A/D converter specifications in circuit designs.
- 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 characteristics of D/A converters and their implementations to include definitions for codes, least-significant bit (LSB), and most-significant bit (MSB); linearity, differential linearity, offset and gain errors; weighted resistor D/A converter; R/2R resistor ladders and D/A converters; weighted current source converters; delta-sigma converters.
- Explain characteristics of A/D converters and their implementations to include definitions for codes, LSB, MSB, and missing codes; linearity, differential linearity, offset and gain errors; counting converter; successive approximation; 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.
- Compare 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.
- Explain how to measure empirically the performance of an algorithm.
- 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:
- 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.
- Explain the speedup attainable in theory and the reasons attainment is often difficult.
- Explain limitations to scalability.
- Discuss parallel algorithm structure and give examples.
- Discuss ways to manage algorithmic execution in multiple threads.
- Describe how to measure 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.
- Explain why heuristics are important in the context of intractable problems.

CE-CAL Supplementary Knowledge Units

CE-CAL-9  Scheduling algorithms
Supplementary

Elective Learning Outcomes: (Note: Some scheduling topics are also included as core topics in the CE-SRM Knowledge Area)
- 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.
- Explain how to analyze the impact of the scheduling algorithm on system performance; give examples.

CE-CAL-10  Basic computability theory
Supplementary

Elective Learning Outcomes:
- Describe how finite state machines represent behavior.
- 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. Number representation and 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.
- Describe how computer engineering uses or benefits from computer architecture and organization.

CE-CAO-2 Relevant tools, standards and/or engineering constraints
Minimum core coverage time: 1 hour
Core Learning Outcomes:
- Discuss the type of information contained in the IEEE 754 standard for floating-point arithmetic.
- Describe and contrast two hardware description languages, such as VHDL and Verilog, and identify tools to simulate computer systems at different levels of design abstraction: system, ISP, register-transfer (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, power, etc.

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.
- Explain 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 programs in assembly language to demonstrate an understanding of machine-level operations.
- Implement some fundamental high-level programming constructs at the assembly-language level, including subroutines.
- Write small programs to access simple input/output devices using program-controlled and interrupt-driven methods.

Elective Learning Outcomes:
- 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:
- Understand 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 compare 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 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 floating-point numbers to and from binary format, using a 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.
- Compare and discuss 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 overcome 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, 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.
- Describe 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 Outcomes:
- Understand how errors in memory systems arise, and describe 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:
Describe 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.

Discuss the use of direct memory access (DMA) to interact with I/O devices.

Determine tradeoffs between program-controlled I/O, interrupt-driven I/O, 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:**
- Describe the characteristics of one or more computer system expansion buses, and 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.
- Describe storage subsystems: storage technologies, storage controllers.
- Describe display subsystems: audio/video subsystems, display controllers.
- Describe input device subsystems (e.g., keyboard, mouse).
- Describe communication subsystems: network controllers, serial and parallel communication functions.

**CE-CAO-10  Multi/Many-core architectures**

*Minimum core coverage time: 6 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: 3 hours*

**Core Learning Outcomes:**
- Explain the differences and tradeoffs between different distributed system paradigms and their usefulness and applicability.
- Discuss 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.
- Understand how the client-server model works in a decentralized fashion.
- Understand 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 the concept of logical clocks versus physical clocks and show how they affect implementation of distributed systems.
- Be familiar with simple election and mutual exclusion algorithms and their applicability.
- Describe approaches to design for parallelism, synchronization, thread safety, concurrent data structures.
- Discuss distributed transactions: models, classification, 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. Analysis and design of digital systems using control+datapath model composed of combinational and sequential building blocks, 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 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), character encoding (ASCII, Unicode), HDL standards (e.g., IEEE 1364 and IEEE 1076), etc.
- 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.
- Convert signed/unsigned, integer/fixed-point decimal numbers to/from representation in the IEEE-754 floating-point standard.
- 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.
- Discuss the physical properties of logic gates such as fan-in, fan-out, propagation delay, power consumption, logic voltage levels, noise margin, etc. and their impact on the constraints and tradeoffs of a design.
- Discuss 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, encoders, etc.
- Describe and design the structure/operation of arithmetic building blocks such as adders (ripple-carry), subtractor, shifters, comparators, etc.
- Describe and design structures for improving adder performance such as carry lookahead, carry select, etc.
- 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.
- Describe 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, shift registers, etc.
- 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, EEPROM, etc.
- Describe the characteristics of dynamic memories.

Elective Learning Outcomes:
- Describe techniques (e.g., handshaking) of asynchronous design, and discuss its advantages (e.g., performance/power in some cases) and design issues (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 discuss the need for control to sequence operations on a datapath.
- Describe and 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/Implement a control+datapath design and generate/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, clock-to-out delay, etc.

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, etc.).

CE-DIG-10  System design and constraints

Minimum core coverage time: 5 hours

Core Learning Outcomes:
- Compare and 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.
- 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:
- Describe the need for systematic testing methods in digital design.
- Define fault models such as stuck-at, bridging, delay, etc.
- Define the terms controllability, observability, test coverage, and test generation when designing a method for testing a digital system.
- Describe different approaches for design for testability such as ad-hoc, full-scan/partial scan, built-in-self-test (BIST), etc.
- 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 (i.e., UART, SPI, I2C, CAN, etc.)
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 (Bluetooth, 802.15.4) and long-range (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) for a target embedded system.
- Compare and 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:
- Compare and 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).
- Compare and 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 duplex and half-duplex communication.
- Compare and contrast parallel I/O versus serial I/O tradeoffs in terms of throughput, wiring cost, 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, encoding method, etc.
- 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, delta-sigma, etc. and give tradeoffs such as conversion time and circuit complexity.
- Demonstrate numerical conversion from a physical quantity such as pressure, temperature, acceleration, etc. 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, light etc. 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, hierarchical state machines, etc.
- Describe techniques used in Real Time Operating Systems such as message passing, preemptive versus cooperative scheduling, semaphores, queues, tasks, co-routines, mutexes, etc.
- 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, clock frequency management, etc. and appropriate GPIO configurations during sleep/hibernate.
- Describe wakeup mechanisms such as watchdog timer, real time clock, external interrupts, etc.
- 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, 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, connectivity.
- Compare and contrast hardware options for adding wireless connectivity to an embedded system such as external smart modules or software stack-plus-radio integrated circuits.
- Compare and 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 topics

**Minimum core coverage time: 3 hours**

**Core Learning Outcomes:**
- Discuss concepts used in I/O buses such as master/slave devices, arbitration, transactions, priorities, packets, etc.
- Compare and 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, data transfers, etc. for one or more advanced serial bus protocols such as the Controller Area Network, Universal Serial Bus, IEEE Firewire, etc.
- Discuss architectures and applications of persistent storage for embedded systems, such as flash drives, SDcards, FRAM, etc.

### 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.
- Compare and contrast performance, power, 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, the TCP/IP model, etc.
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:

- Know the origin and development history of computer network and important applications of computer network.
- Point out figures who makes main contributions and specify what they contributed
- Explain the basic composition and hierarchy of computer network and discourse the role of hierarchy in computer network construction
- Explain the main protocols and the key technologies in computer network

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

Minimum core coverage time: 1 hour

Core Learning Outcomes:

- Describe the broad taxonomy of and wireless standards, for example, cellular network standards vs. 802 family standards.
- Provide an overview of the IEEE 802 family standards, including IEEE802.3, 802.11, 802.15, 802.16. etc.
- Provide an overview of cellular network standards including 2G, 3G, 3.5G, 4G, and LTE.
- Provide an overview of the Bluetooth wireless technology standard.
- Explain functions and basic usages of NS2.
- 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.
- Explain the concept of network architecture and a network’s hardware components.
- Define the elements of a protocol and 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.
- Explain the pros and cons of network topologies such as mesh, star, tree, bus, ring, and 3-D torus.
- Compare and contrast connection-oriented and connectionless services.
- Describe the TCP/IP model, compare and contrast it with OSI model.

CE-NWK-4 Local and wide area networks

Minimum core coverage time: 4 hours

Core Learning Outcomes:

- Understand the basic concepts of LAN and WAN technologies, topologies, and associated tradeoffs.
- Demonstrate understanding of different components and requirements of network protocols and their tradeoffs.
- Explain the definitions and functions of the physical layer and understand features of different transmission media and technologies.
- Demonstrate understanding of basic concepts of error detection and correction at the data-link layer.
- Define circuit and packet switching
- List and explain access and control methods of common shared media.
- Explain key innovations of Ethernet and Gigabit Ethernet.
- Design and build a simple network by designing and implementing a simple network protocol that operates at the physical and data-link layers of the OSI model.
- Explain the key concepts of carrier-sense multiple-access networks (CSMA)
Elective Learning Outcomes:
- Describe protocols for addressing, congestion control, 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, and its development, such as IEEE802.11 a/b/g/n series standards.
- Define the basic concepts in cellular network, such network architecture, framework and LTE etc.
- Describe the main characteristics of mobile IP and explain how it differs from standard IP with regard to mobility management and location management; illustrate how traffic is routed using mobile IP.
- Describe features of typical wireless MAC protocols.

Elective Learning Outcomes:
- Explain wireless CSMA/CA and RTS/CTS enhancement mechanisms.

CE-NWK-6  Network protocols
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Compare and contrast connection-oriented and connectionless services.
- Define a network protocol as specifying 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.
- List some common protocol suites and the services they provide (e.g., IPv4, IPv6, and TCP/UDP).
- Understand functions of the network layer and networking technology.
- Explain architectures and the key technologies of routers and understand the main routing algorithms.
- Understand the concepts of congestion control and the related algorithms.
- Describe main contents of the IP, TCP and UDP protocols.

CE-NWK-7  Network applications
Minimum core coverage time: 2 hours

Core Learning Outcomes: (Note: virtual private networks (VPNs) are included as core topics in the CE-SEC Knowledge Area)
- Describe the key components of a web solution stack using LAMP as an illustrative example.
- Explain the different roles and responsibilities of clients and servers for a range of possible applications.
- Select a range of tools that will ensure 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 (Linux, Apache HTTP server, MySQL, PHP/Perl/Python).
- Describe characteristics of web servers such as handling permissions, file management, and capabilities of common server architectures.
- Describe support tools for web site creation and web management.
- Describe at a high level how 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

Elective Learning Outcomes:
- Know the objectives and main instruments of network management.
- Know the common network management protocols, such as ICMP, SNMP, etc.

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.
- Discuss the issues involved with A/D and D/A conversion in data communications.
- Define 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:
- Define performance metrics.
- Describe how each affects a particular network and/or service paradigm.
- Describe service paradigms such as connection-oriented service and connectionless service.
- Define 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 (web browsing, phone call, TV broadcast, call to a call center).
- Apply some fundamental modeling theory to analyze the performance of a network, for example an 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.
- Give an example of a real application of WSNs.
- Circuit switching vs packet switching: virtual circuit switching(MPLS)
CE-PFP 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 a multidisciplinary 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 a skill in managing projects in the computer engineering field

CE-PFP Core Knowledge Areas

CE-PFP-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 between ethical and legal issues as related to computer engineering.
- Indicate some reasons for studying social and professional issues.
- Identify stakeholders in an issue and our 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-PFP2 Relevant tools, standards, and/or engineering constraints
Minimum core coverage time: 1 hour

Core Learning Outcomes:
- Interpret the social context of a particular 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-PFP-3 Effective communication strategies
Minimum core coverage time: 1 hour

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.
- Learn to build rapport with an audience.
- Develop strategies for effective communication in writing and is speaking.
- Know ways in which body language affects communication.
- Use appropriate visual aids for effective communication.
- Use visualization skill in presenting a technical paper.
- Write technical reports according to specified guidelines.
- Use a presentation mode appropriate for wide range of audiences.
- Use a writing style appropriate to a wide range of audiences.
- Know how to deal with audience questions.
- Write technical reports that are well organized and structured according to accepted standards.

CE-PFP-4 Multidisciplinary team approaches
Minimum core coverage time: 1 hour

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

CE-PFP-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.
- Define ethical relativism and describe some engineering problems related to it.
- Describe the differences between scientific and philosophical approaches to computer engineering dilemmas.
- Recognize 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.
- Explore non-Western philosophical approaches and thought processes as they apply to the computer engineering field.

CE-PFP-6  Engineering solutions and societal effects
Minimum core coverage time: 2 hours

Core Learning Outcomes:
- Recognize the importance of product safety when designing computer systems.
- Describe the differences between correctness, reliability, and safety.
- Explain the limitations of testing as a means to ensure correctness.
- Describe other societal effects beyond risk, safety, and reliability.
- Recognize unwarranted assumptions of statistical independence of errors.
- Discuss the potential for hidden problems in reuse of existing components.
- Explain how computer engineers would assess and manage risk, and how they would inform the public of risk.
- Explain why 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-PFP-7  Professional and ethical responsibilities
Minimum core coverage time: 4 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.
- Explain how 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 arise in software development and determine how to address them technically and ethically.
- Develop a computer use policy with enforcement measures.

CE-PFP-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 how patent and copyright laws might vary internationally.
- Outline the historical development of software patents and compare with other forms of intellectual property protection for software.
- Distinguish among employees, contractors, and consultants and the implications of each status.
- Explore a patent related to computer engineering and provide a summary of its content.
- Explain product and professional liability and their applicability within computer engineering.

CE-PFP-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 nation.
- Discuss how these concepts vary 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.
- Explain 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-PFP-10 Business and management issues**

*Minimum core coverage time: 3 hours*

**Core Learning Outcomes:**
- Describe the assessment of total job costs.
- Evaluate the risks of entering one’s own business.
- 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.
- Explain how computer engineers would use engineering economics in dealing with finances.
- 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.
- Apply engineering economic principles when considering fiscal arrangements.
- Summarize the rationale for antimonopoly efforts.
- Describe two pricing strategies one might use in the computing domain.
- Suggest and defend ways to address limitations on access to computing.

**CE-PFP Supplementary Knowledge Units**

**CE-PFP-11 Tradeoffs in professional practice**

*Supplementary*

**Elective Learning Outcomes:**
- Indicate some important tradeoffs an 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.
- Describe several ways in which shortages in the labor supply affect the information technology industry.
- 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.
- Suggest and 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 and misuse cases.
3. A 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, overview, and principles
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.
- 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.
- Explain and defend the use of each of various security principles, for example 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, for example 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.
- 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 and exploitation
Minimum core coverage time: 4 hours

Core Learning Outcomes:
- Define misuse cases and explain its role in information security.
- 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 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 DES, 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 as long as 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.
- State key properties of secure hash algorithms and families such as SHA-2, SHA-3, or their successors.
- 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 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-SET Strategies for Emerging Technologies
[10 core hours]

Area Scope
The knowledge units in this area collectively encompass the following:
1. Identification and effects of conceptual emerging technologies on computer engineering
2. Identification and effects of applied emerging technologies on computer engineering
3. Exploration of new inventions that have yet to emerge as viable technologies

CE-SET Core Knowledge Units

CE-SET-1 History and overview
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Indicate some reasons for studying strategies for emerging technologies.
- Discuss two emerging technical areas that could flourish within five to ten years.
- Identify some contributors to emerging technologies and relate their achievements to this knowledge area.
- Discuss examples of companies that have failed because they didn’t adapt to emerging technologies.
- Describe how computer engineering benefits from emerging technologies.
- Discuss key historical enablers of new technologies (e.g., underlying technologies, improved manufacturing, new design methods and tools, emerging application areas).
- Describe positive effects/consequences in developing emerging technologies.
- Describe negative effects/consequences in developing emerging technologies.

CE-SET-2 Relevant tools, standards, and/or engineering constraints
Minimum core coverage time: 2 hours

Core Learning Outcomes:
- Describe the role of the computer engineer in an era of emerging technologies.
- Identify stakeholders associated with some emerging technologies.
- Identify strategic assumptions for developing and applying emerging technologies.
- Speculate some engineering constraints and standards surrounding emerging technologies.
- Mention some assumptions and values related to emerging technologies.
- Identify some governmental strategies that would address emerging technologies.
- Indicate ways in which standards would evolve in dealing with emerging technologies.

CE-SET-3 Conceptual emerging technologies
Minimum core coverage time: 2 hours

Core Learning Outcomes:
- Identify ways in which a computer engineer would design environments involving augmented reality and virtual worlds.
- Describe ways in which big data and data analytics might affect the work of a computer engineer.
- Identify some design constraints needed to address computational biology and bioinformatics.
- In an era of machine learning and intelligent systems, explain the role of a computer engineer.
- Discuss engineering strategies needed in developing a culture of green computing and sustainability.
- Identify safety issues in which software-defined networks affect the field of computer engineering.

CE-SET-4 Applied emerging technologies
Minimum core coverage time: 2 hours

Core Learning Outcomes:
- Describe the manner in which a 3D printer would reproduce a miniature model sailboat in a bottle.
- Describe the challenges one would face in designing and producing integrated circuits using 3D printing.
- Explain ways in which nanotechnology can transform the workspace of a computer engineer.
- Describe the internet of things (IoT) and its effect on computer engineering.
- Explain ways in which open source products could affect the practice of a computer engineer.
- Evaluate the future of optical, biological, and quantum computing in the context of a practicing computer engineer.
- Describe the challenges a computer engineer might face in designing a robotic system.
- Explain ways in which Web 2.0 and social networks might affect the field of computer engineering.

CE-SET-5 Additional emerging technologies
Minimum core coverage time: 1 hour

Core Learning Outcomes:
- Identify two conceptual emerging technologies not mentioned in CE-SET-4.
Describe ways in which these conceptual technologies might affect computer engineering.
Identify two applied emerging technologies not mentioned in CE-SET-5.
Describe ways in which these applied technologies might affect computer engineering.

CE-SET Supplementary Knowledge Units

CE-SET-6 Tradeoffs in strategies for emerging technologies
Supplementary

Elective Learning Outcomes:
- Indicate some important topic areas such as quantum computing or 3D printing where tradeoffs become a strategic component for success.
- Articulate some financial tradeoffs in becoming an entrepreneur.
- Describe some unethical tradeoffs that might occur in an emerging technology.
- Discuss emerging technologies where cost tradeoffs affect professional practice.
- Defend ways to address the tradeoffs encountered in distributing free smartphones to third-world nations.
CE-SGP Signal Processing  
[30 core hours]

Area Scope  
The knowledge units in this area collectively encompass the following:  
1. The 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.

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.
  - Indicate some important topic 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.
  - Mention 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 to 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:
- Analyze the frequency response of a system using the DTFT and the DFT.
- Understand frequency selective filters in the z-transform domain.
- Frequency response of discrete time systems
- Poles and zeros in the z-plane
- Implications of assumptions of repeated time series

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.
- State the relationship between time and frequency domains with respect to sampling.
- Explain when spectra are discrete vs. continuous.
- Explain 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:
- Describe 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.
- Apply the sinc function to interpolation and reconstruction.

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.
- Apply a Laplace- to z-domain filter transformation method 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 why window functions are important to digital signal processing.
- Explain how window functions improve transform properties.
- Explain the periodic assumption in spectral analysis.
- Discuss the tradeoffs of common window functions such as rectangular, Blackman, Hann, and Hamming.

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 ways in which one can minimize image noise.
- Describe how digital techniques enhance speech and audio signals.
- Explain how digital techniques cancel noise in audio processing.
- Explain the motivation for audio coding and state key elements of MPEG or related algorithms.
- State some common speech and audio enhancement algorithms.
- Apply an adaptive noise cancellation algorithm.
CE-SPE Systems and Project Engineering
[30 core hours]

Area Scope
The knowledge units in this area collectively encompass the following:
1. 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 of 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

CS-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 that 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.

CS-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 a given area of computer system development (for example, management, modeling, or testing).
- Demonstrate the capability to use a range of tools to support the development of a computer-based system of medium size.
- Indicate 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.

CS-SPE-3 Project management
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Describe basic elements of project management to support development of computer-based systems for a variety of applications, including multidisciplinary 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.
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.

CS-SPE-4 Human-computer interaction (HCI)
Minimum core coverage time: 3 hours

Core Learning Outcomes:
- Summarize some common human-computer interaction styles, and discuss how one would analyze human interaction with computer-based systems.
- Indicate several functional and usability requirements that can be beneficial in developing human-centered computer systems, including users with different abilities (e.g., age, physical disabilities, etc.).
- Identify several fundamental principles for effective GUI design, relevant to different applications and different system platforms in computer engineering.
- Demonstrate an awareness of the capabilities, as well as the limitations, of available input/output and visualization techniques and technologies.
- Discuss the tradeoffs involved when developing a HCI system environment.
- Identify system components that are suitable for the realization of high-quality multimedia interfaces.

Elective Learning Outcomes:
- Design, prototype, and conduct a usability test of a simple 2D GUI, using a provided GUI-builder.
- Evaluate a given 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 and demonstrate this role through the development of a simple application.
- 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.

CS-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.
- Discuss the concept of dependability, and demonstrate an ability to select approaches to the development of a range of computer-based systems that are commensurate with the intended level of dependability.
- 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.
- Discuss the use of failure modes and effects analysis (FMEA) 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.

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

Core Learning Outcomes:
- Recognize the need for a disciplined approach to system development and explain the elements of this in particular contexts.
- Describe the nature of life cycle, the role of lifecycle models, the quality in relation to the life cycle, the influence of system nature, and the size on choice of lifecycle model.
- Describe some common software and hardware development models and show how to apply 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 design for modularity and reuse in the development of a computer-based
system.

Elective Learning Outcomes:
- Select, with justification, system development models most appropriate for the development and maintenance of diverse computer-based systems.

CS-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 (for example, structured analysis or object-oriented-analysis).

CS-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.

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

Core Learning Outcomes:
- Describe some 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.
- Given a system specification, select an appropriate design methodology (for example, structured design or modular design) and create an architectural design of 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.

CS-SPE-10 Concurrent hardware and software design
Minimum core coverage time: 2 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.

Discuss challenges to effective hardware-software co-design, such as demands of hard real-time features.

Demonstrate ability to co-design to achieve particular 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.

CS-SPE-11  System integration, testing and validation
Minimum core coverage time: 2 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.

CS-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.
- 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
- 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
- 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
- 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 the 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 on 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, power consumption, etc. are significant in performance evaluation.
- Explain what to measure relative to 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
- Explain the role of the hypervisor or virtual machine monitor
- Explain what the role of the host machine is and its relationship to the guest machines
- Explain 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
- Explain isolation and security issues arising in virtualized environments
CE-SWD Software Design
[40 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 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 (e.g., procedural), declarative (e.g., functional), and structured (e.g., 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: 4 hours
Core Learning Outcomes: (Note: The CE-CAL Knowledge Area also treats algorithms and overlaps with this Knowledge Unit)
- 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: 4 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 plays
- 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: 2 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 as a result 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)
- Compare 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 and quality**

*Minimum core coverage time: 5 hours*

**Core Learning Outcomes:**
- Demonstrate an understanding of unit testing strategies (white box, black box, grey box, etc.) and tradeoffs
- 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

**CE-SWD-9  Data modeling**

*Minimum core coverage time: 2 hours*

**Core Learning Outcomes:**
- Explain what data model is and how data models are used
- 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 Supplementary Knowledge Units**

**CE-SWD-11  Event-driven and concurrent programming**

*Supplementary*

**Elective Learning Outcomes:**
- Explain the particular 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-12  Using application programming interfaces**

*Supplementary*

**Elective Learning Outcomes:**
- Design reusable functions
- State the purpose of deprecation
- Define backward and forward compatibility problems and some solutions
- Write code against common application program interfaces (APIs) for system services, data structures, network communications, etc.
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 comprehension 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 example curricula that illustrate possible curricula implementations of degree programs that satisfy the required specifications of the body of knowledge detailed in the main body of the report. It contains three- and four-year programs plus an extended five-year program. Although four-year programs are most common, these samples 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 particular institutional needs and priorities.

<table>
<thead>
<tr>
<th>Curriculum</th>
<th>Administrative Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Electrical &amp; Computer Engineering Department</td>
</tr>
<tr>
<td>B</td>
<td>Computer Science Department</td>
</tr>
<tr>
<td>C</td>
<td>Joint – Computer Science and Electrical Engineering Departments</td>
</tr>
<tr>
<td>D</td>
<td>People's Republic of China</td>
</tr>
<tr>
<td>E</td>
<td>Bologna-3</td>
</tr>
<tr>
<td>F</td>
<td>Bologna-5</td>
</tr>
</tbody>
</table>

B.1 Format and Conventions

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

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 ways that identify the curriculum in which it appears and the level at which it appears in the program. Thus, a course numbered MTH,100 is a course in curriculum X commonly taught in the first year (at the freshman level). Likewise, PHY,200 is a course commonly taught in the second year (at the sophomore level); ECE,300 is a course commonly taught in the third year (at the junior level); and course ECE,400 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 U.S. 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 each course a number of semester credit hours, according to the number and types of formal activities within a given week. These are determined as follows.

- Lecture hours: presentation of material in a classroom setting
  - 1 credit hour = One, 1-hour lecture per week
• Laboratory hours: formal experimentation in a laboratory setting
  o 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 last three rows of each table are summary/reference rows. The last table row contains the minimum Core BOK hours for each knowledge area as a reference. The rows labeled Core BOK Units and Supplementary BOK Units lists the knowledge units from each knowledge area covered by this sample curriculum. Since all of 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 of 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 an exhaustive list of topics taught in these courses. A list of the BOK knowledge areas and knowledge units covered by these courses augment these descriptions.

B.2 Preparation to Enter the Profession

The sample curricula in this appendix have as a major goal the preparation of graduates for entry into the computer engineering profession. There are many ways of building 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.

The CE2016 steering committee designed these curricula to ensure appropriate coverage of the core topics of the computer engineering BOK as defined in this report. However, as also discussed in the main report, there are many
other elements to creating a program that will effectively prepare graduates for the professional practice of computer engineering, such as design and laboratory experience, oral and written communication, and usage of modern engineering tools. Accordingly, professional accreditation addresses more than just curriculum, and readers interested in accreditation should consult the relevant criteria from 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 (Curriculum 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 United States of America. The remaining sample curricula (Curriculum D, E, F) illustrate typical programs in computer engineering that one might find in China, the United Kingdom, Europe, Asia, and other parts of the world. Two of the curricula are consistent with the Bologna Declaration, with the three-year program leading to a typical bachelor of engineering degree and the five-year program leading to a typical master’s of engineering degree.

The common requirements spread widely across a range of courses and allow revisiting the subject matter with spiral learning taking place. Each curriculum contains sufficient flexibility to support various areas of specialization. Each program structure allows a broadly based course of study and provides selection from among many professional electives. In all cases, the 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. It 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. The thorough preparation afforded by the computer engineering curriculum includes the broad education necessary to understand the impact of engineering solutions in a global and societal context.

Graduates of each program should be well prepared for professional employment or advanced studies. They should understand the various areas of computer engineering such as applied electronics, digital devices and systems, software design, and computer architectures, systems, and networks. Graduates should be able to apply their acquired knowledge and skills to these and other areas of computer engineering. They will also possess design skills and have a deep understanding of hardware issues, software issues, models, the interactions between these issues, and related applications. The thorough preparation afforded by this computer engineering curriculum includes the broad education necessary to understand the impact of engineering solutions in a global and societal context.
B.4 Curriculum 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 in this appendix), with more hours devoted to computer engineering topics. 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 coverage in software topics to enable graduates to be effective computer engineers.

B.4.2 Summary of Requirements

This program of study contains 5 required computer science courses (17 credits) and 14 required electrical and computer engineering (ECE) courses (42 credits). Flexibility is provided by the two computer engineering elective courses (6 credits), which can be chosen from courses in either the computer science or electrical and computer engineering departments. 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. This curriculum requires 39 courses, with credit hours distributed as follows.

<table>
<thead>
<tr>
<th>Credit-hours</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Mathematics</td>
</tr>
<tr>
<td>10</td>
<td>Natural science (Physics, Chemistry)</td>
</tr>
<tr>
<td>24</td>
<td>English composition, humanities and social sciences</td>
</tr>
<tr>
<td>38</td>
<td>Required electrical and computer engineering</td>
</tr>
<tr>
<td>17</td>
<td>Required computer science</td>
</tr>
<tr>
<td>6</td>
<td>Technical electives (from ECE or CSC)</td>
</tr>
<tr>
<td>4</td>
<td>Design project</td>
</tr>
<tr>
<td>0</td>
<td>Free electives</td>
</tr>
<tr>
<td></td>
<td><strong>120</strong> <strong>TOTAL Credit Hours for Computer Engineering Program</strong></td>
</tr>
</tbody>
</table>
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 tech electives: approved elective course in either department

<table>
<thead>
<tr>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester 1</td>
<td></td>
<td></td>
<td>Semester 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTH 101</td>
<td>Calculus I</td>
<td>3</td>
<td>MTH 101</td>
<td>Calculus II</td>
<td>3</td>
</tr>
<tr>
<td>CHM 101</td>
<td>Chemistry I &amp; Lab</td>
<td>4</td>
<td>PHY 101</td>
<td>Physics I</td>
<td>3</td>
</tr>
<tr>
<td>CSC 101</td>
<td>Programming I &amp; Lab</td>
<td>4</td>
<td>CSC 102</td>
<td>Programming II &amp; Lab</td>
<td>4</td>
</tr>
<tr>
<td>English Composition I</td>
<td></td>
<td>3</td>
<td>ECE 101</td>
<td>Introduction to ECE</td>
<td>2</td>
</tr>
<tr>
<td>Humanities Elective</td>
<td></td>
<td>3</td>
<td>English Composition II</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td><strong>Total Credit Hours</strong></td>
<td></td>
<td>17</td>
<td><strong>Total Credit Hours</strong></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Semester 3</td>
<td></td>
<td></td>
<td>Semester 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTH 201</td>
<td>Calculus III</td>
<td>3</td>
<td>MTH 203</td>
<td>Differential Equations</td>
<td>3</td>
</tr>
<tr>
<td>PHY 201</td>
<td>Physics II</td>
<td>3</td>
<td>MTH 204</td>
<td>Discrete Structures</td>
<td>3</td>
</tr>
<tr>
<td>ECE 201</td>
<td>Digital Devices &amp; Lab</td>
<td>4</td>
<td>ECE 202</td>
<td>Microprocessors &amp; Lab</td>
<td>4</td>
</tr>
<tr>
<td>CSC 201</td>
<td>Data Structures</td>
<td>3</td>
<td>ECE 203</td>
<td>Circuits/Electronics I</td>
<td>3</td>
</tr>
<tr>
<td>MTH 202</td>
<td>Linear Algebra</td>
<td>3</td>
<td>MTH 205</td>
<td>Probability &amp; Statistics</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total Credit Hours</strong></td>
<td></td>
<td>16</td>
<td><strong>Total Credit Hours</strong></td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>Semester 5</td>
<td></td>
<td></td>
<td>Semester 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSC 301</td>
<td>Algorithms</td>
<td>3</td>
<td>CSC 302</td>
<td>Client/Server Programming</td>
<td>3</td>
</tr>
<tr>
<td>ECE 301</td>
<td>Circuits/Electronics II &amp; Lab</td>
<td>4</td>
<td>ECE 303</td>
<td>Signals &amp; Systems</td>
<td>3</td>
</tr>
<tr>
<td>ECE 302</td>
<td>Digital System Design &amp; Lab</td>
<td>3</td>
<td>ECE 304</td>
<td>Data Communication</td>
<td>3</td>
</tr>
<tr>
<td>Humanities Elective</td>
<td></td>
<td>3</td>
<td>ECE 305</td>
<td>Computer Architecture</td>
<td>3</td>
</tr>
<tr>
<td>Social Science Elective</td>
<td></td>
<td>3</td>
<td>Social Science Elective</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td><strong>Total Credit Hours</strong></td>
<td></td>
<td>13</td>
<td><strong>Total Credit Hours</strong></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Semester 7</td>
<td></td>
<td></td>
<td>Semester 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECE 401</td>
<td>CE Design I</td>
<td>2</td>
<td>ECE 402</td>
<td>CE Design II</td>
<td>2</td>
</tr>
<tr>
<td>ECE 403</td>
<td>Embedded Systems &amp; Lab</td>
<td>3</td>
<td>ECE 404</td>
<td>Computer Security</td>
<td>3</td>
</tr>
<tr>
<td>ENG 401</td>
<td>Writing for Engineers</td>
<td>3</td>
<td>ECE 405</td>
<td>Operating Systems</td>
<td>3</td>
</tr>
<tr>
<td>CE Elective</td>
<td></td>
<td>3</td>
<td>CE Elective</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Fine Arts Elective</td>
<td></td>
<td>3</td>
<td>Social Science Elective</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>14</td>
<td><strong>Total</strong></td>
<td></td>
<td>14</td>
</tr>
</tbody>
</table>
### 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 | C    | A    | A    | D    | E    | A    | O    | G    | S    | W    | Y    | P    | E    | S    | E    | T    | S    | P    | E    | R    | W    | D    | A    | C    | D    | L    | A    | P    |
|-----------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| CSCA101   | 1-4  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| CSCA102   |      | 4-8  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| CSCA201   |      |      | 5-9  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| CSCA301   |      |      |      | 1-8  |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| CSCA302   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA101   |      |      |      |      | 1-3,5|      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA201   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA202   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA203   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA301   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA302   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA303   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA304   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA305   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA401   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA402   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA403   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA404   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| ECEA405   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| MTH 202   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| MTH 203   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| MTH 204   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| MTH 205   |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Core BOK  | 1-10 | 1-11 | 1-8  | 1-11 | 1-13 | 1-11 | 1-11 | 1-11 | 1-11 | 1-11 | 1-11 | 1-11 | 1-6  | 1-7  | 1-7  | 1-12 | 1-6  | 1-10 | 1-7  | 1-9  | 1-10 | 1-10 |      |      |      |      |
| Units Covered | 50  | 60  | 30  | 50  | 40  | 20  | 20  | 20  | 10  | 30  | 30  | 20  | 40  | 30  | 30  | 20  | 30  | 20  | 30  |      |      |      |      |
| Supplementary BOK Units |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Core BOK Hours (min) |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
| Core BOK Hours (min) |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
B.4.5 Curriculum A – Course Summaries

**CSCA101: 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

**CSCA102: Intermediate Computer Programming**
Object-oriented problem solving, design, and programming; introduction to data structures, algorithm design and complexity
Prerequisite: CSCA101
Credit Hours: 4 Lecture Hours: 42  Lab Hours: 42
*BOK Coverage*: CE-SWD 4-8

**CSCA201: 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: CSCA102 and College Algebra or equivalent
Credit Hours: 3 Lecture Hours: 42  Lab Hours: 0
*BOK Coverage*: CE-SWD 5-9

**CSCA301: 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: CSCA201, MTH 201 and MTHA204
Credit Hours: 3 Lecture Hours: 42  Lab Hours: 0
*BOK Coverage*: CE-CAL 1-8

**CSCA302: 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: CSCA201
Credit Hours: 3 Lecture Hours: 28  Lab Hours: 42
*BOK Coverage*: CE-SWD 10, CE-SPE 1-12; Supplementary CE-SWD 12

**ECEA101: 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-PFP 1-3, 5

**ECEA201: 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 CSCA101
Credit Hours: 4 Lecture Hours: 42  Lab Hours: 42
*BOK Coverage*: CE-DIG 1-9

**ECEA202: Microprocessors**
Architecture of microprocessor-based systems; study of microprocessor operation, assembly language, arithmetic operations, and interfacing
Prerequisite: ECEA201, CSCA201
Credit Hours: 4 Lecture Hours: 42  Lab Hours: 42
*BOK Coverage*: CE-ESY 1-8, CE-CAO 3, 5, CE-DIG 3
ECEA203: 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

ECEA301: 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: ECEA203
Credit Hours: 4 Lecture Hours: 42 Lab Hours: 42
BOK Coverage: CE-CAE 5-10

ECEA302: Digital System Design
Modern digital design techniques using logic synthesis, hardware description languages; field programmable gate arrays, and modular building blocks
Prerequisite: ECEA202
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 42
BOK Coverage: CE-DIG 1, 2, 6-11

ECEA303: 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: ECEA301
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0
BOK Coverage: CE-SGP 1-7

ECEA304: 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: ECEA202
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0
BOK Coverage: CE-NWK 1-11, CE-SET 3-5

ECEA305: 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: ECEA202
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0
BOK Coverage: CE-CAO 1-11

ECEA401: CE Design I
Lectures on teaming, project management, engineering standards, economics, and ethical and professional issues. Student must select faculty mentor, perform project design, and present orally
Prerequisite: ECEA302, CSCA302
Credit Hours: 2 Lecture Hours: 14 Lab Hours: 45
BOK Coverage: CE-SPE 7-10, CE-PFP 1-6, 11

ECEA402: CE Design II
Lectures on teaming, project management, engineering standards, economics, and ethical and professional issues. Student must select faculty mentor, perform project design, and present orally
Prerequisite: ECEA401
Credit Hours: 3 Lecture Hours: 14 Lab Hours: 42
BOK Coverage: CE-SPE 10-12, CE-PFP 7-11, CE-SET 1-6

ECEA403: Embedded Systems
Advanced topics in embedded systems design using contemporary practice; interrupt-driven, reactive, real-time, object-oriented, and distributed client/server embedded systems
Prerequisite: ECEA302, ECEA301
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
ECEA404: 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: ECEA304
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0
BOK Coverage: CE-SEC 1-11, CE-SET 3-5

ECEA405: 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: ECEA202
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

MTHA204: Discrete Structures
Concepts of algorithms, induction, recursion, proofs, topics from logic, set theory, combinatorics, graph theory fundamental to study of computer science
Prerequisite: CSCA101 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 would sponsor this B.S. 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 but not a large engineering program. 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 topics. For this reason, the CE2016 steering committee designed several courses specifically to provide coverage of the core topics of the CE BOK. In particular, we designed courses that cover the traditional EE topics in the CE core to cover the core material without providing significant breadth or depth beyond the core topics. In contrast, a number of the computer science courses do go somewhat beyond the core material. There is, however, sufficient coverage of the EE, hardware, and systems topics to enable graduates to be effective as 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 from electrical engineering, comprising 42 credit hours of study. The program achieves flexibility through a judicious choice of three technical electives and a culminating project. The computer engineering segment of the curriculum, including technical electives and capstone, comprises 51 credit-hours of study. 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 (six credits) to ensure that all graduates have significant design experience, as well as experience with teamwork and modern engineering tools. We assume 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. It requires 39 courses, with credit hours distributed as follows.

<table>
<thead>
<tr>
<th>Credit-hours</th>
<th>Topic areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Mathematics and statistics</td>
</tr>
<tr>
<td>11</td>
<td>Natural science (physics, chemistry)</td>
</tr>
<tr>
<td>33</td>
<td>Humanities, social sciences, composition, and literature</td>
</tr>
<tr>
<td>27</td>
<td>Required computer science (excluding design project)</td>
</tr>
<tr>
<td>11</td>
<td>Required electrical engineering</td>
</tr>
<tr>
<td>9</td>
<td>Technical electives (from computer science or engineering)</td>
</tr>
<tr>
<td>5</td>
<td>Culminating design project (from computer science)</td>
</tr>
<tr>
<td>6</td>
<td>Free electives</td>
</tr>
<tr>
<td><strong>120</strong></td>
<td><strong>TOTAL Credit Hours for Computer Engineering Program</strong></td>
</tr>
</tbody>
</table>
### B.5.3 Four-Year Model for Curriculum B

<table>
<thead>
<tr>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester 1</td>
<td></td>
<td></td>
<td>Semester 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTH 101</td>
<td>Calculus I</td>
<td>3</td>
<td>MTH 102</td>
<td>Calculus II</td>
<td>3</td>
</tr>
<tr>
<td>CHM 101</td>
<td>Chemistry I</td>
<td>4</td>
<td>PHYS 101</td>
<td>Physics I</td>
<td>4</td>
</tr>
<tr>
<td>CSC 101</td>
<td>Computer Science I</td>
<td>4</td>
<td>CSC 102</td>
<td>Computer Science II</td>
<td>4</td>
</tr>
<tr>
<td>English Comp.</td>
<td></td>
<td>3</td>
<td>Humanities / Social Science</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total Credit Hours</td>
<td>14</td>
<td></td>
<td>Total Credit Hours</td>
<td>14</td>
</tr>
</tbody>
</table>

| Semester 3   |                        |        | Semester 4   |                                    |        |
| MTH 201      | Differential Equations | 3      | MTH 203      | Linear Algebra                     | 3      |
| PHY 201      | Physics II             | 3      | CSC 201      | Algorithm Design                   | 3      |
| MTH 202      | Discrete Structures    | 3      | ELE 202      | Circuits and Electronics           | 4      |
| ELE 201      | Introduction to Digital Design | 4 | CSC 202 | 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 302      | Embedded Systems                   | 3      |
| CSC 301      | Computer Architecture  | 3      | CSC 303      | Computer Networks & Information Security | 3    |
| ELE 301      | Systems and Signal Processing | 3   | CSC 304      | 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 401      | Senior Project I       | 2      | CSC 402      | Senior Project II                  | 3      |
| CSC 403      | Ethics and Professionalism | 1    | Technical elective | 3      |
| Technical elective | 3            | 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.

<table>
<thead>
<tr>
<th>BOK Area</th>
<th>Course</th>
<th>Core BOK Units Covered</th>
<th>Supplementary BOK Units</th>
<th>Core BOK Hours (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1-10 1-11 1-10 1-12 1-8 1-9 1-11 1-5 1-8 1-12 1-6 1-10 1-9 1-10 1-9</td>
<td>10 11 7-8 11-12</td>
<td>50 30 60 50 40 20 20 20 10 30 30 20 40 30 30 20 30</td>
</tr>
<tr>
<td>BOK</td>
<td>Course</td>
<td>1-8 1-11 1-8 1-11 1-8 1-9 1-11 1-5 1-8 1-12 1-6 1-10 1-9 1-10 1-9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>I</td>
<td>C</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>G</td>
<td>E</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>Y</td>
<td>S</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>K</td>
<td>W</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>S</td>
<td>E</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>T</td>
<td>G</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>P</td>
<td>E</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>M</td>
<td>S</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>W</td>
<td>D</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>F</td>
<td>C</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>L</td>
<td>A</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>P</td>
<td>S</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>E</td>
<td>C</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>S</td>
<td>E</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>T</td>
<td>G</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>P</td>
<td>E</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>M</td>
<td>S</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>W</td>
<td>D</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>F</td>
<td>C</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>L</td>
<td>A</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
<tr>
<td>P</td>
<td>S</td>
<td>1-8, 6-8</td>
<td>1-5, 6-10, 11-12</td>
<td>1-5 30</td>
</tr>
</tbody>
</table>
B.5.5 Curriculum B – Course Summaries

**CSCB101: 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: 4 Lecture Hours: 42 Lab Hours: 42
*BOK Coverage:* CE-SWD 1-5

**CSCB102: Computer Science II**
Second course in programming languages and systems; topics include object-oriented design, data structures, recursion, data modeling, fundamental concepts in software engineering
Prerequisites: CSCB101
Credit Hours: 4 Lecture Hours: 42 Lab Hours: 42
*BOK Coverage:* CE-SWD 1, 6-10; Supplementary CE-SWD 11-12

**CSCB201: Algorithm Design**
Analysis and design of algorithms, with topics to include algorithm design strategies, searching and sorting algorithms, parallel algorithms, tradeoffs in algorithmic performance, algorithmic complexity
Prerequisites: CSCB102
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0
*BOK Coverage:* CE-CAL 1-8; Supplementary CE-CAL 10

**CSCB202: Computer Organization**
Introductory course in computer organization and architecture; topics include processor organization, instruction set architecture, memory system organization, performance, and interfacing fundamentals
Prerequisites: CSCB101
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0
*BOK Coverage:* CE-CAO 1-4, 6-8

**CSCB301: 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: CSCB202
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0
*BOK Coverage:* CE-CAO 6-11; CE-DIG 8, 10

**CSCB302: 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: CSCB202, CSCB301
Credit Hours: 3 Lecture Hours: 28 Lab Hours: 42
*BOK Coverage:* CE-ESY 1-12

**CSCB303: 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: CSCB202, MTHB202; Co-Requisite CSCB304
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0
*BOK Coverage:* CE-NWK 1-8; CE-SEC 1-11

**CSCB304: 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: CSCB201, CSCB202
CSC 401: 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: CSEB302, CSEB312, CSEB332
Credit Hours: 2 Lecture Hours: 14 Lab Hours: 42
BOK Coverage: CE-PFP 2-4, 6-9, CE-SPE 1-12, CE-SET 1-5

CSC 402 Senior Project II
Continuation of Senior Project I focused upon implementation of a project design
Prerequisites: CSC 401
Credit Hours: 3 Lecture Hours: 14 Lab Hours: 84
BOK Coverage: CE-PFP 2-4, 6-9, CE-SPE 1-12, CE-SET 1-5

CSC 403: 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; topics addressed include the process of ethical decision-making, privacy and confidentiality, computer crime, professional codes and responsibilities, software piracy, the impact of computers on society
Prerequisites: Junior standing
Credit Hours: 1 Lecture Hours: 0 Lab Hours: 42
BOK Coverage: CE-PFP 1, 5, 6-10

ELE 201: 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 101
Credit Hours: 4 Lecture Hours: 42 Lab Hours: 42
BOK Coverage: CE-DIG 1-7, 9, CE-CAO 5; Supplementary CE-DIG 11

ELE 202: 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 301: 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 202
Credit Hours: 3 Lecture Hours: 42 Lab Hours: 0
BOK Coverage: CE-SGP 1-8

MTH 202: 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. CE 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 5 traditional CSC courses (e.g., programming fundamentals), 5 traditionally ECE courses (e.g., circuits & electronics), and 6 CE courses that can be taught by computer engineering faculty members in either department (e.g., computer architecture & organization). Specialization occurs via 18 credits of 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). The credit hours distribution for this curriculum is as follows.

<table>
<thead>
<tr>
<th>Credit-hours</th>
<th>Topic areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Mathematics</td>
</tr>
<tr>
<td>16</td>
<td>Natural science (physics, chemistry)</td>
</tr>
<tr>
<td>18</td>
<td>Humanities, social sciences, composition, literature</td>
</tr>
<tr>
<td>12</td>
<td>Required computer engineering (CE)</td>
</tr>
<tr>
<td>19</td>
<td>Required electrical and computer engineering (ECE)</td>
</tr>
<tr>
<td>16</td>
<td>Required computer science (CSC)</td>
</tr>
<tr>
<td>12</td>
<td>Technical electives</td>
</tr>
<tr>
<td>6</td>
<td>Design project (CPR)</td>
</tr>
<tr>
<td>0</td>
<td>Free electives</td>
</tr>
<tr>
<td>120</td>
<td>TOTAL Credit Hours for Computer Engineering Program</td>
</tr>
</tbody>
</table>
### 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 tech electives: approved elective course in either department

<table>
<thead>
<tr>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semester 1</strong></td>
<td></td>
<td></td>
<td><strong>Semester 2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTH 101</td>
<td>Calculus I</td>
<td>4</td>
<td>MTH 102</td>
<td>Calculus II</td>
<td>4</td>
</tr>
<tr>
<td>CHM 101</td>
<td>Chemistry I + Lab</td>
<td>4</td>
<td>PHY 101</td>
<td>Physics I + Lab</td>
<td>4</td>
</tr>
<tr>
<td>Humanities / Social Science</td>
<td>3</td>
<td>CSC_101</td>
<td>Programming Fundamentals I</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Humanities / Social Science</td>
<td>3</td>
<td>CHM 102</td>
<td>Chemistry II or Biological Science</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Total Credit Hours</strong></td>
<td><strong>14</strong></td>
<td></td>
<td><strong>Total Credit Hours</strong></td>
<td><strong>14</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semester 3</th>
<th>Semester 4</th>
<th>Semester 5</th>
<th>Semester 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTH 201</td>
<td>Calculus III</td>
<td>4</td>
<td>MTH 202</td>
</tr>
<tr>
<td>PHY 201</td>
<td>Physics II + Lab</td>
<td>4</td>
<td>ECE_202</td>
</tr>
<tr>
<td>CSC_201</td>
<td>Programming Fundamentals II</td>
<td>3</td>
<td>CE_201</td>
</tr>
<tr>
<td>ECE_201</td>
<td>Digital &amp; Logic Design + Lab</td>
<td>4</td>
<td>CSC_202</td>
</tr>
<tr>
<td></td>
<td>Humanities / Social Science</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td><strong>Total Credit Hours</strong></td>
<td><strong>15</strong></td>
<td></td>
<td><strong>Total Credit Hours</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semester 7</th>
<th>Semester 8</th>
<th>Semester 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC_401</td>
<td>Operating Systems</td>
<td>3</td>
</tr>
<tr>
<td>ECE_401</td>
<td>Signals &amp; Systems</td>
<td>3</td>
</tr>
<tr>
<td>CE_402</td>
<td>CE Design I</td>
<td>3</td>
</tr>
<tr>
<td>CE Tech Elective*</td>
<td>3</td>
<td>CE_403</td>
</tr>
<tr>
<td>Humanities / Social Science</td>
<td>3</td>
<td>Professional Communication &amp; Ethics for Engineers</td>
</tr>
<tr>
<td><strong>Total Credit Hours</strong></td>
<td><strong>15</strong></td>
<td></td>
</tr>
</tbody>
</table>
B.6.4 Mapping of Computer Engineering BOK to Curriculum C

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

<table>
<thead>
<tr>
<th>BOK Area</th>
<th>Course</th>
<th>Core BOK Units covered</th>
<th>Supplementary BOK Units</th>
<th>Core BOK Hours (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-12 3 8-10</td>
<td>1-10 1-8 1-10 1-8 1-10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-10 1-9</td>
<td>1-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-12 3 8-10</td>
<td>1-10 1-8 1-10 1-8 1-10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-10 1-9</td>
<td>1-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-12 3 8-10</td>
<td>1-10 1-8 1-10 1-8 1-10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-10 1-9</td>
<td>1-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-12 3 8-10</td>
<td>1-10 1-8 1-10 1-8 1-10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-10 1-9</td>
<td>1-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-12 3 8-10</td>
<td>1-10 1-8 1-10 1-8 1-10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-10 1-9</td>
<td>1-9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-12 3 8-10</td>
<td>1-10 1-8 1-10 1-8 1-10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-10 1-8 1-10 1-8 1-10</td>
<td>1-10 1-9</td>
<td>1-9</td>
</tr>
</tbody>
</table>
B.6.5 Curriculum C – Course Summaries

**CSCC101: 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

**CSCC201: 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: CSCC101
Credit Hours: 3 Lecture Hours: 42  Lab Hours: 0
*BOK Coverage*: CE-CAL 4, SWD 3-7; Supplementary SWE 12

**CSCC202: Introduction to Discrete Structures**
Concepts of discrete structures including Boolean algebra, first-order algebra, proof techniques, set theory, and graph theory
Prerequisite: CSCC201, MTH 101
Credit Hours: 3 Lecture Hours: 42  Lab Hours: 0
*BOK Coverage*: CE-DSC 1-9

**CSCC301: Data Structures & 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: CSCC201
Credit Hours: 4 Lecture Hours: 56  Lab Hours: 0
*BOK Coverage*: CE-CAL 1-8, CE-SWE 4-5

**CSCC401: Operating Systems**
Basic operating systems and their components: scheduling, resource management, process management, interrupt handling, concurrent processing, and system performance evaluation
Prerequisite: CSCC301, CEC201
Credit Hours: 3 Lecture Hours: 42  Lab Hours: 0
*BOK Coverage*: CE-SRM 1-6; Supplementary CE-SRM 7

**ECEC201: 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: CSCC101
Credit Hours: 3 Lecture Hours: 42  Lab Hours: 14
*BOK Coverage*: CE-DIG 1-9

**ECEC202: 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

**ECEC301: Embedded & Microprocessor Systems + Lab**
Topics in microprocessor-based embedded systems, synchronous and asynchronous serial communication, interfacing, interrupt handling, data acquisition, real-time processing
Prerequisite: CEc201, ECEc201
Credit Hours: 4 Lecture Hours: 42  Lab Hours: 14
*BOK Coverage*: CE-CAO 6-8, CE-ESY 1-8

**ECEC302: Digital System Design + Lab**
Advanced topics in digital design: modular design of combinational and sequential logic building blocks, control-
Computing Curriculum  
Interim Curriculum Report  
Computer Engineering 2016  
Version: 2015 October 25  

datapath; extensive use of hardware description language (HDL) and programmable logic devices (e.g., FPGAs);  
system architecture design and evaluation  
Prerequisite: ECEC201, CEc 201  
Credit Hours: 4 Lecture Hours: 42  Lab Hours: 14  
BOK Coverage: CE-DIG 2, DIG 6-10, SPE 8-9, SET 4-5  

ECEC401: Signals & Systems  
Continuous-time and discrete-time signal analysis including Fourier series and discrete-time and discrete 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  

CEC201: 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: CSCC201, ECEC201  
Credit Hours: 3 Lecture Hours: 42  Lab Hours: 0  
BOK Coverage: CE-CAO 1-9, CE-LAL 6  

CEC301: 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: CEC201, CSCC301  
Credit Hours: 3 Lecture Hours: 42  Lab Hours: 0  
BOK Coverage: CE-SPE 1-12, PFP 2, PDP 6, SRM 3, SWD 8-10  

CEC302: 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: CEC201, ECEC301  
Credit Hours: 3 Lecture Hours: 42  Lab Hours: 0  
BOK Coverage: CE-CAL 6, NWK 1-8, SEC 8  

CEC401: Computer and Network Security  
Concepts in computer and network security including data security and integrity, vulnerabilities and exploitation,  
cryptography, authentication, network and web security, and trust computing  
Prerequisite: CEC201, CEc 302  
Credit Hours: 3 Lecture Hours: 42  Lab Hours: 0  
BOK Coverage: CE-NWK 8, SEC 1-11, SET 4-5  

CEC402: 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: CEC301, ECE 301  
Credit Hours: 3 Lecture Hours: 14  Lab Hours: 84  
BOK Coverage: CE-PFP 1-4, SET 1-4  

CEC403: 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: CEC402  
Credit Hours: 3 Lecture Hours: 14  Lab Hours: 84  
BOK Coverage: CE-PFP 2-10, SET 2-6
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. This type of department would administer this 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 24 required computer science and technology courses. The program provides flexibility by allowing 30 hours’ elective courses chosen from courses in either the electrical engineering or the computer science and technology departments. Lab hours occur in five courses giving 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. This curriculum requires 55 courses, with credit hours distributed as follows.

<table>
<thead>
<tr>
<th>Credit-hours</th>
<th>Topic areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Mathematics</td>
</tr>
<tr>
<td>10</td>
<td>Natural science (physics, chemistry)</td>
</tr>
<tr>
<td>6</td>
<td>Elective mathematics and natural Science</td>
</tr>
<tr>
<td>42</td>
<td>Humanities, social sciences, composition, literature, foreign language</td>
</tr>
<tr>
<td>24</td>
<td>Required computer science and engineering (CE)</td>
</tr>
<tr>
<td>12</td>
<td>Labs or computer science and engineering (CE)</td>
</tr>
<tr>
<td>30</td>
<td>Technical electives – basic and advance (CE)</td>
</tr>
<tr>
<td>15</td>
<td>Capstone project for graduation (CE)</td>
</tr>
<tr>
<td>0</td>
<td>Free electives</td>
</tr>
<tr>
<td><strong>160</strong></td>
<td>TOTAL Credit Hours for Computer Engineering Program</td>
</tr>
</tbody>
</table>
B.7.3 Four-Year Model for Curriculum D

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 tech electives: approved elective course in either department

<table>
<thead>
<tr>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester 1</td>
<td>Semester 2</td>
<td>Semester 3</td>
<td>Semester 4</td>
<td>Semester 5</td>
<td>Semester 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical Education I</td>
<td>1</td>
<td>Physical Education II</td>
<td>1</td>
<td>CST_D 301</td>
<td>Practice in English</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Humanities and social I</td>
<td>3</td>
<td>Humanities and social II</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English language I</td>
<td>2</td>
<td>English language II</td>
<td>2</td>
<td>PHY 201</td>
<td>Physics I</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTH 101</td>
<td>Calculus I</td>
<td>5</td>
<td>MTH 201</td>
<td>Calculus II</td>
<td>5</td>
<td>PHY 401</td>
<td>Physics II</td>
<td>4</td>
</tr>
<tr>
<td>MTH 102</td>
<td>Linear Algebra I</td>
<td>4</td>
<td>MTH 202</td>
<td>Linear Algebra II</td>
<td>2</td>
<td>PHY 402</td>
<td>Lab of Physics I</td>
<td>1</td>
</tr>
<tr>
<td>MTH 103</td>
<td>Discrete Mathematics I</td>
<td>3</td>
<td>CST_D 201</td>
<td>Fundamentals Object-oriented Programming</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CST_D 101</td>
<td>Fundamentals of Programming</td>
<td>3</td>
<td>MTH 203</td>
<td>Discrete Mathematics II</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CST_D 102</td>
<td>Introduction of Information Science</td>
<td>1</td>
<td>PHY 201</td>
<td>Physics I</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elective of Humanities</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total Credit Hours</strong></td>
<td>23</td>
<td><strong>Total Credit Hours</strong></td>
<td>22</td>
<td><strong>Total Credit Hours</strong></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHY 401</td>
<td>Physics II</td>
<td>4</td>
<td>CST_D 501</td>
<td>Digital Logic Circuits &amp; Lab</td>
<td>4</td>
<td>CST_D 601</td>
<td>Practice in Assembly Language</td>
<td>3</td>
</tr>
<tr>
<td>PHY 402</td>
<td>Lab of Physics I</td>
<td>1</td>
<td>PHY 501</td>
<td>Lab of Physics II</td>
<td>1</td>
<td>CST_D 602</td>
<td>Practice in Java</td>
<td>2</td>
</tr>
<tr>
<td>CST_D 401</td>
<td>Data Structures</td>
<td>4</td>
<td>MTH 501</td>
<td>Probability and Statistics</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Humanities and social III</td>
<td>5</td>
<td>CST_D 502</td>
<td>Formal Language and Automata</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>English language III</td>
<td>2</td>
<td>CST_D 503</td>
<td>Introduction to Artificial Intelligence</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Physical Education III</td>
<td>1</td>
<td>Physical Education IV</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTH 401</td>
<td>Introduction to Functions of One Complex Variable</td>
<td>2</td>
<td>English language IV</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CST_D 402</td>
<td>Fundamentals of Electronics &amp; Lab</td>
<td>4</td>
<td>Elective of humanities and social</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CST_D 504</td>
<td>Introduction to High Performance Computing</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total Credit Hours</strong></td>
<td>23</td>
<td><strong>Total Credit Hours</strong></td>
<td>18</td>
<td><strong>Total Credit Hours</strong></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semester 7</td>
<td>Semester 8</td>
<td>Semester 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------</td>
<td>--------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSTD 701 Computer Organization</td>
<td>CSTD 801 Operating System</td>
<td>CSTD 901 Design of Whole Computer System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSTD 702 Computer Network</td>
<td>CSTD 802 Computer Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSTD 703 Principle of Compiler System</td>
<td>CSTD 803 Numerical analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSTD 704 Software Engineering</td>
<td>Politics Theory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSTD 705 Principle of Signal Processing</td>
<td>Humanities and social IV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elective of humanities and social</td>
<td>CSTD 804 Emerging technology on Software</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSTD 706 Introduction of VLSI Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total Credit Hours</strong> 19</td>
<td><strong>Total Credit Hours</strong> 18</td>
<td><strong>Total Credit Hours</strong> 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semester 10</th>
<th>Semester 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humanities and social IV</td>
<td>CSTD 1101 Capstone project for graduation</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>CSTD 1001 Network Security and Management</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CSTD 1002 Embedded System</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td><strong>Total Credit Hours</strong> 11</td>
<td><strong>Total Credit Hours</strong> 15</td>
</tr>
</tbody>
</table>
### 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) | C | A | A | E | L | C | O | G | Y | K | P | E | S | C | T | P | E | R | M | A | F | D | S | L | P |
| CST_D101         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D102         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D201         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D301         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D401         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D402         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D501         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D502         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D503         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D504         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D601         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D602         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D701         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D702         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D703         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D704         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D705         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D706         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D801         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D802         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D803#        |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D804         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D901         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D1001        |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D1002        |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CST_D1101        |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTH101           |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTH102           |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTH103           |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTH201           |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTH202           |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTH203           |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTH401#          |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTH501           |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Core BOK Units   | 1-10 | 1-8 | 1-11 | 1-10 | 1-12 | 1-8 | 1-10 | 1-11 | 1-11 | 1-5 | 1-8 | 1-12 | 1-6 | 1-10 | 1-7 | 1-9 | 1-10 | 1-9 |
| Covered          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Supplementary    | 12 | 10 | 13 | 9-10 | 11 | 7 | | | | | | | | | | | | | | | |
| BOK Units       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Hours (min)      | 50 | 60 | 30 | 50 | 40 | 20 | 20 | 20 | 10 | 30 | 30 | 20 | 40 | 30 | 30 | 20 | 30 | 30 | 30 | 20 | 30 | 30 | 20 | 30 |

# this course does not cover any BOK Unit.
B.7.5 Curriculum D – Course Summaries

**CSTD101 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: 48 Lecture Hours: 48 Lab Hours: 0
- BOK Coverage: CE-SWD 1-6

**CSTD102 Introduction of Information Science**
Introductory course in information science; topics include the fundamental problems and solving methods including some emerging technologies in information science
- Prerequisite: None
- Credit Hours: 16 Lecture Hours: 16 Lab Hours: 0
- BOK Coverage: CE-SET 1, 3-4

**CSTD201 Fundamentals Object-oriented Programming**
This course covers the basics of C++ programming, object-oriented programming; introductory problem solving and computer programming using object-oriented techniques
- Prerequisite: CSTD101
- Credit Hours: 32 Lecture Hours: 32 Lab Hours: 0
- BOK Coverage: CE-SWD 1-2, 6-8

**CSTD301 Practice in Programming**
This highly practical course aims at students’ innovative ideas and hands-on skills; through teaching and experiments, students are able to grasp general knowledge about C language and basic programming algorithms
- Prerequisite: CSTD101
- Credit Hours: 32 Lecture Hours: 16 Lab Hours: 32
- BOK Coverage: CE-SWD 3-8

**CSTD401 Data Structures**
This course covers 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: CSTD101
- Credit Hours: 64 Lecture Hours: 64 Lab Hours: 32
- BOK Coverage: CE-CAL 1-8, 10

**CSTD402 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: 64 Lecture Hours: 48 Lab Hours: 16
- BOK Coverage: CE-CAE 1-10, 12

**CSTD501 Digital Logic Circuits & Lab**
This course emphasizes on the analysis, design, and implementation of the digital logic circuit. Topics include Boolean algebra and logic simplification, design and analysis of combinational circuits and sequential circuits, competition and risk management in logic design
- Prerequisite: CSTD402
- Credit Hours: 64 Lecture Hours: 48 Lab Hours: 16
- BOK Coverage: CE-DIG 1-10

**CSTD502 Formal Language and Automata**
This course mainly focuses on the formal languages, context free languages, and the related computation models. It also discusses Turning machine and the foundations of theoretical computing.
- Prerequisite: MTH103, MTH203
- Credit Hours: 32 Lecture Hours: 32 Lab Hours: 0
BOK Coverage: None

CSTp503 Introduction to Artificial Intelligence
Introductory course on artificial intelligence, topics include fundamental problems in artificial intelligence, production system, search problem, AND/OR search graph, predicate calculus, advanced search, expert systems.
Prerequisite: CSTp101, MTH103, MTH203
Credit Hours: 32 Lecture Hours: 32 Lab Hours: 0

CSTp504 Introduction to High Performance Computing
The course covers the following topics: 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: CSTp101
Credit Hours: 32 Lecture Hours: 32 Lab Hours: 0
BOK Coverage: CE-CAO1-2, 10-11

CSTp601 Practice in Assembly Language
Practice course on the basic theory, programming tools, programming methods and applications of assembly language
Prerequisite: CSTp101
Credit Hours: 48 Lecture Hours: 32 Lab Hours: 32
BOK Coverage: CE-CAO 3, 8-9

CSTp602 Practice in Java
Practice course on Java programming, topics include Java’s history and evolution, development tools, object-oriented programming methods and network programming technologies
Prerequisite: CSTp101
Credit Hours: 48 Lecture Hours: 32 Lab Hours: 32
BOK Coverage: CE-SWD 4, 8

CSTp701 Computer Organization
Introduction to single CPU computer organization and architecture; topics include instruction set architecture, datapath and controller, pipelining and performance measurement, hierarchical memory organization, bus structures, I/O interfacing and communications, peripheral equipment
Prerequisite: CSTp101, CSTp501
Credit Hours: 64 Lecture Hours: 48 Lab Hours: 16
BOK Coverage: CE-CAO 1-9, DIG 8

CSTp702 Computer Network
Computer network architecture, typical protocols and standards of LAN, WAN, wireless and mobile networks, classical network applications, network management and security; topics about networks social effects, network engineer’s professional and responsibilities are also included
Prerequisite: CSTp101, CSTp401
Credit Hours: 48 Lecture Hours: 48 Lab Hours: 0
BOK Coverage: CE-NWK 1-10, CE-PFP 6-7 CE-SEC 8-9

CSTp703 Principle of Compiler System
Covers the following topics: 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: CSTp401, CSTp502, CSTp602,
Credit Hours: 32 Lecture Hours: 32 Lab Hours: 0
BOK Coverage:

CSTp704 Software Engineering
Focusing on the software develop methods, the topics include software life cycle, design methods, software testing and performance evaluation, software development management
Prerequisite: CSTp101, CSTp401
Credit Hours: 48 Lecture Hours: 48 Lab Hours: 0
BOK Coverage: CE-SPE 1-5,7-9 CE-PFP 3-5,10
CSTD705 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: 48 Lecture Hours: 48 Lab Hours: 0  
BOK Coverage: CE-SGP 1-8

CSTD706 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: CSTD101, CSTD401  
Credit Hours: 32 Lecture Hours: 32 Lab Hours: 0  
BOK Coverage: CE-DIG 9-11

CSTD801 Operating System  
Introduction to the computer operating system, topics include process management, process scheduling, storage management, file management and device management  
Prerequisite: CSTD101, CSTD401, CSTD701  
Credit Hours: 48 Lecture Hours: 48 Lab Hours: 0  
BOK Coverage: CE-SRM 1-7

CSTD802 Computer Architecture  
This course focuses on the structure, design principles, and other key concepts in computer architecture; topics include 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: CSTD501, CSTD701, CSTD703  
Credit Hours: 48 Lecture Hours: 48 Lab Hours: 0  
BOK Coverage: CE-CAO 1-2, 10-11 CE-SET 4

CSTD803 Numerical analysis  
This course covers the following main topics: (1) numerical computation and error analysis, (2) direct and iteration methods to solve system of linear equations, calculate Eigen values and Eigen vectors of matrices, (3) function approximation and interpolation method: Lagrange's interpolation, Newton’s interpolation, subsection low interpolation, and cubic spline interpolation, (4) approximation for numerical integration and differential, (5) approximation method for solving equations  
Prerequisite: MTH101, MTH102, MTH201, MTH202  
Credit Hours: 48 Lecture Hours: 48 Lab Hours: 0  
BOK Coverage:  

CSTD804 Emerging technology on Software  
This course introduces the frontier researches of software technology, which include system software, data and knowledge engineering, software engineering; computer aided design technology  
Prerequisite:  
Credit Hours: 32 Lecture Hours: 32 Lab Hours: 0  
BOK Coverage: CE-SET 1-5

CSTD901 Design of Whole Computer System  
Practice course on the design and implement of whole computer system, students are required to design and implement a mini-computer that has the basic hardware, operating system and compiler in groups of two or three person  
Prerequisite: CSTD701, CSTD702, CSTD703, CSTD801  
Credit Hours: 160 Lecture Hours: 20 Lab Hours: 140  
BOK Coverage: CE-SPE 1-12

CSTD1001 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, network measure, management, and operation  
Prerequisite: CSTD702  
Credit Hours: 48 Lecture Hours: 48 Lab Hours: 0  
BOK Coverage: CE-SEC 1-11, CE-NWK 8
CSTD1002 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: CSTD701, CSTD801
Credit Hours: 48  Lecture Hours: 48  Lab Hours: 0
BOK Coverage: CE-ESY 1-13

CSTD1101 Capstone project for graduation
The capstone project for graduation is in the last semester to ensure that all graduates have significant design experience, as well as experience with teamwork and modern engineering tools
Prerequisite: None
Credit Hours: 240  Lecture Hours: 32  Lab Hours: 400
BOK Coverage: CE-PFP 1-11
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.

To explain, the Bologna Declaration was signed on 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, equivalent to 1500-1800 hours of study in all countries. Typically, a bachelor’s degree requires 180-240 ECTS taking 3-4 years, and a master’s degree 90-120 ECTS 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 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 as illustrated in Appendix B.9.

Students desiring intensive study in computer engineering will find this program to be a challenging and rewarding experience. The curriculum provides a broad foundation in the science and engineering of computers and digital systems with emphasis on theory, analysis, and design. The curriculum will also develop analytical, computer and applied skills which will enable students to analyze, design and test digital and computer systems, architectures, networks, and processes. Graduates of the program will be able to apply and evaluate various areas of computer engineering such as applied electronics, digital devices and systems, electromagnetic fields and waves, computer architectures, systems, and networks. They will be equipped to move to a related second cycle or master’s degree program as described in Appendix B.9. Graduates must possess design skills and they 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.
For the purposes of ease of comparison, this model attempts to outline the degree in conformity with the previous curricula implementations. This means that the curriculum comprises courses having 3 credit hours (or 42 contact hours) of study with approximately ten courses per year.

**B.8.2 Summary of Requirements**

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

In these programs of study, one would expect that some element of laboratory experience would constitute an integral part of each course in computer engineering; the purpose of this integration is to reinforce and illustrate the work of the associated lectures. In some classes, the amount of laboratory work would typically be heavier than in other parts. Accordingly, we have adopted the following convention:

- where intensive laboratory activity is desirable, a 3-credit class is typically composed of 28 hours of lectures and 28 hours of laboratory work plus associated recitation time
- where less intensive laboratory activity is desired, then typically 14 hours of laboratory work and 42 hours of lecture work is required together with the associated recitation hours

The 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.

<table>
<thead>
<tr>
<th>Credit-hours</th>
<th>Topic areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Mathematics</td>
</tr>
<tr>
<td>0</td>
<td>Natural science (physics, chemistry)</td>
</tr>
<tr>
<td>0</td>
<td>Humanities, social sciences, composition, literature</td>
</tr>
<tr>
<td>21</td>
<td>Required computer engineering (CE)</td>
</tr>
<tr>
<td>21</td>
<td>Required electrical and computer engineering (ECE)</td>
</tr>
<tr>
<td>15</td>
<td>Required software engineering</td>
</tr>
<tr>
<td>12</td>
<td>Required computer science (CSC)</td>
</tr>
<tr>
<td>6</td>
<td>Technical electives</td>
</tr>
<tr>
<td>0</td>
<td>Design project (CE)</td>
</tr>
<tr>
<td>0</td>
<td>Free electives</td>
</tr>
<tr>
<td><strong>90</strong></td>
<td><strong>TOTAL Credit Hours for Computer Engineering Program Bologna-3.</strong></td>
</tr>
</tbody>
</table>

The illustration allows for electives (options) available in the third year of study. Students may chose two third-year electives selected from a set of four computer engineering courses (modules). The course options are as follows:

*Third-year Electives (computer engineering)*

- Computer Graphics
- Intelligent Systems and Robotics
- Device Development
- 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

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 tech electives: approved elective course in either department

<table>
<thead>
<tr>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semester 1</strong></td>
<td></td>
<td></td>
<td><strong>Semester 2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTHE101</td>
<td>Discrete Structures for Computing</td>
<td>3</td>
<td>MTHE103</td>
<td>Calculus and Geometry</td>
<td>3</td>
</tr>
<tr>
<td>MTHE102</td>
<td>Applied Probability and Statistics</td>
<td>3</td>
<td>MTHE104</td>
<td>Linear Algebra</td>
<td>3</td>
</tr>
<tr>
<td>CSC_E101</td>
<td>Computer and Information Systems</td>
<td>3</td>
<td>CE_E101</td>
<td>Concepts in Computer Engineering</td>
<td>3</td>
</tr>
<tr>
<td>ELE_E101</td>
<td>Foundations of Electronics</td>
<td>3</td>
<td>ELE_E102</td>
<td>Digital Circuits I</td>
<td>3</td>
</tr>
<tr>
<td>SWEE_E101</td>
<td>Programming Basics</td>
<td>3</td>
<td>SWEE_E102</td>
<td>Programming Fundamentals</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semester 3</strong></td>
<td></td>
<td></td>
<td><strong>Semester 4</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTH_E201</td>
<td>Mathematics for Engineers</td>
<td>3</td>
<td>CSC_E202</td>
<td>Operating Systems and Net-Centric</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Computing</td>
<td></td>
</tr>
<tr>
<td>CE_E201</td>
<td>Networking &amp; Communications</td>
<td>3</td>
<td>CE_E203</td>
<td>Professional Issues in Computer</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Engineering</td>
<td></td>
</tr>
<tr>
<td>CSC_E201</td>
<td>Analysis and Design of Algorithms</td>
<td>3</td>
<td>ELE_E202</td>
<td>Analogue Circuits</td>
<td>3</td>
</tr>
<tr>
<td>CE_E202</td>
<td>Computer Organization</td>
<td>3</td>
<td>CE_E204</td>
<td>Computer Systems Engineering</td>
<td>3</td>
</tr>
<tr>
<td>ELE_E201</td>
<td>Digital Circuits II</td>
<td>3</td>
<td>CE_E205</td>
<td>Embedded Computer Systems</td>
<td>3</td>
</tr>
</tbody>
</table>

**Total Credit Hours** 15  

<table>
<thead>
<tr>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
<th>Course</th>
<th>Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semester 5</strong></td>
<td></td>
<td></td>
<td><strong>Semester 6</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE_E301</td>
<td>Individual Project I</td>
<td>3</td>
<td>CE_E302</td>
<td>Individual Project II</td>
<td>3</td>
</tr>
<tr>
<td>CSC_E301</td>
<td>Programming Languages and Syntax</td>
<td>3</td>
<td>CE_E303</td>
<td>Computer Architecture</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Directed Tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELE_E301</td>
<td>Signals and Systems</td>
<td>3</td>
<td>ELE_E302</td>
<td>System Control</td>
<td>3</td>
</tr>
<tr>
<td>SWEE_E301</td>
<td>Software Engineering</td>
<td>3</td>
<td>ELE_E303</td>
<td>Digital Signal Processing</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td></td>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>

**Total Credit Hours** 15  

---

For Review and Comment
### 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 | C | A | A | D | E | S | W | F | P | S | E | G | P | S | R | W | D | A | C | D | L | P | R |
| CEE101    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CEE201    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CEE202    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CEE203    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CEE204    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CEE205    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CEE303    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CSCe101   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CSCe201   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CSCe202   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CSCe301   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ELEe101   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ELEe102   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ELEe201   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ELEe202   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ELEe301   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ELEe302   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| ELEe303   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTHe101   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTHe102   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTHe103   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTHe104   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| MTHe201   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| SWEe101   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| SWEe102   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| SWEe301   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| **Core BOK Units Covered** | 1-10 | 1-8 | 1-11 | 1-10 | 1-12 | 1-8 | 1-10 | 1-11 | 1-5 | 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 | 6 | 9-10 | 7-8 | 11-12 | 10 |
| **Core BOK Hours (min)** | 50 | 30 | 60 | 50 | 20 | 20 | 20 | 20 | 10 | 30 | 30 | 20 | 40 | 30 | 30 | 30 | 30 | 30 |
B.8.5 Curriculum E – Course Summaries

**CEE101: 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
CE2016 BOK Coverage: CE-SEC 1-5, CE-SPE 1-5

**CEE201: 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
CE2016 BOK Coverage: CE-NWK 1-8

**CEE202: 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: CEE101
Credit hours: 3  Lecture hours: 42  Lab hours: 14
CE2016 BOK Coverage: CE-CAO 1-7

**CEE203: Professional Issues in Computer Engineering**
Critical examination of ethical problems associated with computer engineering; discussion of these problems conducted within the framework of classical philosophical ethical theories; legal and quasi-legal (i.e., policy and regulative) issues; topics addressed include the process of ethical decision-making, privacy and confidentiality, computer crime, professional codes and responsibilities, professional practice, system security and more generally cyber security, impact of computers on society
Prerequisites: Second-year standing
Credit hours 3  Lecture hours: 42  Lab hours: 0
CE2016 BOK Coverage: CE-PFP 1-10

**CEE204: 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: CEE201
Credit hours: 3  Lecture hours: 28  Lab hours: 28
CE2016 BOK Coverage: CE-SEC 6-10, CE-SPE 6-10

**CEE205: Embedded Computer Systems**
Nature of embedded systems, particular problems, special issues; role in computer engineering; embedded microcontrollers, embedded software; real time systems, problems of timing and scheduling; testing and performance issues, reliability; low power computing, energy sources, leakage; design methodologies, software tool support for development of such systems; problems of maintenance and upgrade; networked embedded systems
Prerequisites: Two courses in calculus and two courses in physics
Credit hours: 3  Lecture hours: 28  Lab hours: 28
CE2016 BOK Coverage: CE-ESY 1-12
**CE301: Individual Project I**

Comprehensive project spanning two semesters; students undertake an individual project which involves addressing a significant technical problem they embark on this under the guidance of a supervisor; student are expected to demonstrate an ability to apply the disciplined approaches of the course in addressing the solution to the problem; students produce a final thesis on the work and this together with a demonstration of the working system will form the assessment

Prerequisites: Third-year standing  
Credit hours: 3 Lecture Hours 42  Lab Hours: 14  
CE2004 BOK Coverage: Project dependent

**CE302: Individual Project II**

Continuation of Individual Project I  
Prerequisites: Third-year standing  
Credit hours: 3 Lecture hours: 0  Lab hours: 42  
CE2016 BOK Coverage: Project dependent

**CE303: 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: CE202  
Credit hours: 3 Lecture hours: 42  Lab hours: 14  
CE2016 BOK Coverage: CE-CAO 8-11

**CSC101: 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  
CE2016 BOK Coverage: CE-NWK 1-3, CE-CAL 1-3, SPE 1-4, CE-SRM 1-3

**CSC201: 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: MTH101, SWE101  
Credit hours: 3 Lecture hours: 28  Lab hours: 28  
CE2016 BOK Coverage: CE-CAL 1-10

**CSC202: 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: CSC101  
Credit hours: 3 Lecture hours: 42  Lab hours: 14  
CE2016 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
CE2016 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, Thévenin’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
CE2016 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
CE2016 BOK Coverage: CE-CAE 1-6

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
CE2016 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
CE2016 BOK Coverage: CE-CAE 4-8, CE-SET 1-5

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
CE2016 BOK Coverage: CE-SGO 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
ELE 303: 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 301
Credit hours: 3  Lecture hours: 42  Lab hours: 14
CE2016 BOK Coverage: CE-SGP 7-8

MTH 101: 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
CE2016 BOK Coverage: CE-DSC 1-9

MTH 102: 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
CE2016 BOK Coverage: CE-PRS 1-9

MTH 103: 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 the use of graphics packages
Prerequisites: Two courses in calculus (achieved before entering the program)
Credit hours: 3  Lecture hours: 42  Lab hours: 14
CE2016 BOK Coverage: CE-ACF 1-7

MTH 104: 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 101
Credit hours: 3  Lecture hours: 42  Lab hours: 14
CE2016 BOK Coverage: CE-LAL 1-10

MTH 201: 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 102, MTH 103, MTH 104
Credit hours: 3  Lecture hours: 42  Lab hours: 14
CE2016 BOK Coverage:

SWE 101: 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
CE2016 BOK Coverage: CE-SWD 1-7

SWE\textsubscript{E}102: 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\textsubscript{E}101
Credit hours: 3 Lecture hours: 28 Lab hours: 28
CE2016 BOK Coverage: CE-CAL 1-8

SWE\textsubscript{E}301: 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\textsubscript{E}101, SWE\textsubscript{E}102
Credit hours: 3 Lecture hours: 28 Lab hours: 28
CE2016 BOK Coverage: CE-SWD 8-12.
B.9 Curriculum F: Bologna-5 Model

Computer Engineering Program Representative of a Second Cycle Program within Bologna Declaration

B.9.1 Program Goals and Features

This curriculum model demonstrates a typical program in computer engineering as one might find in the second cycle within the Bologna arrangements; the latter has a summary in Appendix B.8. Second cycle programs are essentially two-year programs that reach the level equivalent to a masters of engineering degree. At that level, depth is necessary as well as an element of specialization. In some countries, the expectation is that almost all students achieve this level of attainment whereas in others perhaps 30%-40% or less of students advance to this level.

Students embarking on this program will have already completed a three-year degree program such as that outlined in Appendix B.8. Graduates of this two-year second cycle program will be well versed in various areas of computer engineering at an advanced level and will generally be equipped to commence employment in a variety of possible settings, with the better students being ready to undertake research. They will have a deep 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. The program maintains the ethos surrounding computer engineering.

For the purposes of ease of comparison, this model attempts to outline the degree program in conformity with the previous curricula implementations. The curriculum comprises courses each having 3 credit hours (or 42 contact hours) of study with approximately ten courses per year. As befits a master’s degree, the emphasis on (normally individual) project work especially in the final year is 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.

B.9.2 Summary of Requirements

This model curriculum builds upon and extends the first cycle model described in Appendix B.8. At this level the implementation of this model exposes the student to a substantial influx on new and innovative subject matter representing the current activities and developments in computer engineering. Such an approach allows students to develop specializations or specialty tracts with the intent of having a focused orientation of their studies. It also indicates that degrees in computer engineering can have different emphases and individual students can follow different specializations.

The expectation remains that some element of laboratory experience would constitute an integral part of each course in computer engineering; the purpose of this integration is, as before, to reinforce and illustrate the work of the associated lectures. In some classes, the amount of laboratory work would typically be heavier than in other parts. Again, 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. Typically, 14 hours of laboratory work and 42 hours of lecture work are required together with the associated recitation hours. The project in the final year is a substantial piece of work, taking students to the frontiers of knowledge and possibly introducing them to well defined research endeavors.

The illustration of a program in computer engineering using this model appears as four semesters, the equivalent of 20 courses and 60 credit hours of study. The illustration allows for electives (options) available in both years of study. The precise set of electives will depend on faculty expertise. A selection of possible electives follows.

Possible Electives

- Computer Graphics and Multimedia
- Intelligent Systems and Robotics
- Fault Tolerant Systems Design
- Hardware Software Co-design

Page 133 of 142
Machine Learning          Network Security
Robotics and Simulation   Software for Telecommunications
Cyber security           Interactive Systems Design
Biologically Inspired Computing   Medical Information Systems
Advanced Computer Design Control Systems Engineering
Grid Computing           Safety Critical Systems
B.9.3 Five-Year Model for Curriculum F

What follows is a possible curriculum model for a second cycle degree program. Details of the courses (or modules) appear in the course summaries in section B.9.5.

CE: offered in the computer engineering department  
ELE: offered in the electrical engineering department  
CSC: offered in the computer science department

<table>
<thead>
<tr>
<th>Course Description</th>
<th>Credit</th>
<th>Course Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester 7</td>
<td></td>
<td>Semester 8</td>
<td></td>
</tr>
<tr>
<td>CEF701</td>
<td>3</td>
<td>ELEF702</td>
<td>3</td>
</tr>
<tr>
<td>Business and Economics of Computer Engineering</td>
<td></td>
<td>Sensors and Actuators</td>
<td></td>
</tr>
<tr>
<td>CEF702</td>
<td>3</td>
<td>CEF703</td>
<td>3</td>
</tr>
<tr>
<td>Internet Programming and Protocols</td>
<td></td>
<td>Mobile Computing</td>
<td></td>
</tr>
<tr>
<td>CSCF701</td>
<td>3</td>
<td>CSCF702</td>
<td>3</td>
</tr>
<tr>
<td>Concurrency and Parallelism</td>
<td></td>
<td>Intelligent Systems</td>
<td></td>
</tr>
<tr>
<td>ELEF701</td>
<td>3</td>
<td>CEF704</td>
<td>3</td>
</tr>
<tr>
<td>Audio and Vision Systems</td>
<td></td>
<td>High Performance Computing</td>
<td></td>
</tr>
<tr>
<td>Option A</td>
<td>3</td>
<td>Option B</td>
<td>3</td>
</tr>
<tr>
<td>Total Credit Hours</td>
<td>15</td>
<td>Total Credit Hours</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester 9</td>
<td></td>
</tr>
<tr>
<td>CEF801</td>
<td>6</td>
</tr>
<tr>
<td>Senior Project I</td>
<td></td>
</tr>
<tr>
<td>CEF802</td>
<td>3</td>
</tr>
<tr>
<td>Ethics and Professionalism</td>
<td></td>
</tr>
<tr>
<td>CEF803</td>
<td>6</td>
</tr>
<tr>
<td>Senior Project II</td>
<td></td>
</tr>
<tr>
<td>Option C</td>
<td>3</td>
</tr>
<tr>
<td>Total Credit Hours</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course Description</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester 10</td>
<td></td>
</tr>
<tr>
<td>CEF804</td>
<td>3</td>
</tr>
<tr>
<td>Ubiquitous and Pervasive Computing</td>
<td></td>
</tr>
<tr>
<td>CEF805</td>
<td>6</td>
</tr>
<tr>
<td>Senior Project II</td>
<td></td>
</tr>
</tbody>
</table>

B.9.4 Mapping of Computer Engineering BOK to Curriculum F

The Bologna-5 sample has no knowledge unit mappings because the first three years are the same as Bologna-3, the equivalent undergraduate portion, to which all knowledge units already mapped. See CE Curriculum E for related details.

B.9.5 Curriculum F – Course Summaries

**CEF701: Business and Economics of Computer Engineering**

Engineering and uncertainty, engineering processes, strategies, proposals, decision making; economic concepts, utility, value, cost, consumers; supply and demand; initial costs, maintenance, fixed, variable, and marginal costs; interest rates, simple and compound interest; money value, past, present, and future values; cash flow; present and future worth, payback periods.

Prerequisites: Fourth-year standing  
Credit hours: 3 Lecture hours 42 Lab hours: 14

**CEF702: Internet Programming and Protocols**

Internet architectures, media signal processing, electronics for home entertainment; routing and forwarding, reliable data recovery, mobility, social networking; software on the internet, building applications, accessing databases; interface considerations
Prerequisites: Fourth-year standing  
Credit hours: 3  Lecture hours 28  Lab hours: 28

**CE4703: Mobile Computing**
Bluetooth hardware and development kits; signal processing requirements for high-speed wireless data communications, communications system-on-a-chip, video over mobile networks, pen-based user interfaces; low power design; security considerations  
Prerequisites: Fourth-year standing  
Credit hours: 3  Lecture hours 42  Lab hours: 14

**CE4704: High Performance Computing**
Aims of high-performance-computing; important application areas; architectural considerations; high performance clusters, multi-core; circuits for high-performance input and output; parallel I/O systems; performance analytics; workload management, scheduling issues; approaches to HPC programming; use of the cloud and issues related to this  
Prerequisites: Fourth-year standing  
Credit hours: 3  Lecture hours 28  Lab hours: 28

**CE4801: Senior Project I**
Students undertake a substantial project that builds on the work of the program and takes students to the frontiers of computer engineering. This may involve exploring research ideas but should typically necessitate the development of a prototype. A review of the relevant and up-to-date literature should be involved. The analysis of the performance of the prototype should be part of the project as well as a formal presentation of the final work. A member of staff should act as supervisor and should provide relevant guidance to the student.  
Prerequisites: Fourth-year standing  
Credit hours: 6  Lecture hours 3  Lab hours: 100

**CE4802: Project Management in Computer Engineering**
Project initiation, management, and success, appraisal and risk, quality systems and implementation, environmental impacts; contracts, costs, finance, planning, organization, personnel management; contract strategies and policy; turnkey operations, global issues, international commerce, negotiation, customs, and law  
Prerequisites: Fourth-year standing  
Credit hours: 6  Lecture hours 3  Lab hours: 100

**CE4803: Senior Project II**
Continuation of Senior Project I  
Prerequisites: Fourth-year standing  
Credit hours: 6  Lecture hours 3  Lab hours: 100

**CE4804: Entrepreneurship and the Engineer**
Basic concepts of marketing, business organization, management accounting, business finance, and financial feasibility analysis of new business ventures and of new project proposals in established firms; appreciate the financial risks and rewards; strategies for investing in new ventures; entrepreneurial strategies, venture development processes, bringing products from the idea to market and operation; business planning, implementation, operation, and success, business plans, organization, budgets, accounting methods and processes, capital and debt, business analyses  
Prerequisites: Fourth-year standing  
Credit hours: 3  Lecture hours 42  Lab hours: 0

**CE4805: Ubiquitous and Pervasive Computing**
Current trends towards universal presence of mobile computing, computer networks, and wireless communication; how network devices are aware of their environment; identification of current status, fundamental issues, future problems and applications; current research topics in the area of ubiquitous and pervasive computing; design issues; integration and processing of sensor-based input; wireless infrastructures; security and user-interfaces; integrated, multimodal input and output and application areas  
Prerequisites: Fourth-year Standing  
Credit hours: 3  Lecture hours 42  Lab hours: 14

**CSC4701: Concurrency and Parallelism**
Parallel architectures including multi-core; considerations of cache coherence; parallel programming models; methodologies; parallel algorithms; limitations of parallelism; parallel programming performance  
Prerequisites: Fourth-year standing  
Credit hours: 3  Lecture hours 42  Lab hours: 14
CSC702: Intelligent Systems
Intelligent systems in a computer engineering context, with a focus on examples such as intelligent transportation systems, driverless cars, navigation systems, etc.; nature of intelligence and the implications for software, its reliability; feature recognition; importance of real time reliable information; system considerations in support of intelligent systems in computer engineering; human-computer partnership considerations
Prerequisites: Fourth-year standing
Credit hours: 3 Lecture hours 42  Lab hours: 14

CSC801: Ethics and Professionalism
Social context of computer engineering; need for public protection; intellectual property; privacy and civil liberties; industry standards; professional behavior and communication; levels of safety and security, and being able to justify these; the economics of computer engineering; trade-offs and justifying decisions; legal considerations and the global context; professional considerations including the role of professional societies and professional development; MOOCs and their role in ongoing professional development
Prerequisites: Fourth-year standing
Credit hours: 3 Lecture hours 42  Lab hours: 14

ELE701: Audio and Vision Systems
Audio processing fundamentals; speech enhancement; noise cancellation, speech recognition systems and related applications; vision system fundamentals; smoothing images; object recognition and face recognition; related applications
Prerequisites: Fourth-year standing
Credit hours: 3 Lecture hours 42  Lab hours: 14

ELE702: Sensors and Actuators
Range of sensors and actuators including the range of position sensors, force sensors, vibration sensors, chemical sensors and biosensors; utilization of these in various environments including hazardous environments; programming considerations and applications
Prerequisites: Fourth-year standing
Credit hours: 3 Lecture hours 42  Lab hours: 14
Bibliography

Note that the following represents a comprehensive set of references, which is an extension of references cited in this report.


[ACM] Association for Computing Machinery (ACM), Code of Ethics; <https://www.acm.org/about/code-of-ethics>


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


[INVEST] Investopedia; http://www.investopedia.com/terms/s/soft-skills.asp


[SIT] Society on Social Implications of Technology (SITT) of IEEE; http://ieeessit.org/


Additional Material

This is a placeholder for possible additional material.