Computing Curricula 2001

Computer Engineering

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

Preliminary (Pre-Iron Man) Draft

October 3, 2003
CCCE Task Force Members

Dave Soldan, Chair
Mitch Theys
Joe Hughes
Pradip Srimani
Andrew McGettrick
Victor P. Nelson
Murali Varanasi
Alan Clements
Bob Sloan
Danial J. Neebel
Esther A. Hughes
Bob Klenke
Ron Hoelzeman
Jim Aylor
Gerald Engel
John Impagliazzo
Douglas Lyon
Greg Peterson
Contents

Chapter 1. Introduction

1.1 Overall structure of the CC series
1.2 Overview of the CCCE process
1.3 Structure of the CC Computer Engineering report

Chapter 2. Computer Engineering as a Discipline

2.1 Background
2.2 Evolution of the field
2.3 Characteristics of Computer Engineering graduates
   2.3.1 Curricular Preparation
   2.3.2 Program and student evaluation
2.4 Organizational Options
2.5 Computer Engineering Outside the United States

Chapter 3. Principles

Chapter 4. Overview of the Computer Engineering Body of Knowledge

4.1 Introduction
4.2 Structure of the Body of Knowledge
4.3 Learning Outcomes
4.4 Core and Elective Units
4.5 Units and Time Required for Coverage
4.6 Core Hours and a Complete Program
4.7 Summary of the Body of Knowledge

Chapter 5. Integration of Engineering Practice into the Computer Engineering Curriculum

5.1 The Nature of Computer Engineering
5.2 Design in the Curriculum
   5.2.1 Design Throughout the Curriculum
   5.2.2 The Capstone Design Experience
5.3 The Laboratory Experience
   5.3.1 Laboratory Experiments
   5.3.2 Practical Activity
5.4 The Role of Engineering Tools and Re-use
5.5 Applications of Computer Engineering Principles
5.6 Complementary Skills
5.7 Communication Skills
5.8 Teamwork Skills
5.9 Lifelong Learning Skills
5.10 The Business Perspective
Chapter 6. Professionalism and Computer Engineering

6.1 Introduction
6.2 Decisions in a Societal Context
6.3 Fostering Professional Practice
6.4 Summary

Chapter 7. Curriculum Implementation Issues

7.1 General Considerations
7.2 Basic Computer Engineering Components
   7.2.1 Capstone Project
   7.2.2 Engineering, Professional, Legal, and Ethical Issues
   7.2.3 Assessment of Teaching and Learning
7.3 Courses Outside Computer Engineering
   7.3.1 Mathematical Requirements
   7.3.2 Scientific Method
   7.3.3 Communication Skills
7.4 Degree Program Implementation: Skills and Examples
   7.4.1 Course considerations
   7.4.2 Sample Curricula

Chapter 8. Institutional Challenges

8.1 The Need for Local Adaptation
8.2 Principles for Curriculum Design
8.3 The Need for Adequate Laboratory Resources
8.4 Attracting and Retaining Faculty
8.5 Summary

Chapter 9. Conclusions

Bibliography
Appendix A  Knowledge Units

CE-ALG - Algorithms and Complexity
CE-CAO - Computer Architecture and Organization
CE-CSE - Computer Systems Engineering
CE-CSY - Circuits and Systems
CE-DIG - Digital Logic
CE-DSC - Discrete Structures
CE-DSI - Digital Signal Integrity
CE-DSP - Digital Signal Processing & Multimedia
CE-DSV - Digital System Verification
CE-DSY - Distributed Systems
CE-ELE - Electronics
CE-ESY - Embedded Systems
CE-HCI - Human Computer Interaction
CE-INM - Information Management
CE-INS - Intelligent Systems
CE-NWK - Computer Networks
CE-OPS - Operating Systems
CE-PRF - Programming Fundamentals
CE-PRS - Probability and Statistics
CE-SPR - Social and Professional Issues
CE-SWE - Software Engineering
CE-TAV - Test and Verification
CE-VLS - VLSI/ASIC Design

Appendix B  Sample Curricula
Chapter 1

Introduction

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

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

As indicated in the charter, the goal of the CC effort is to revise Computing Curricula 1991 so that it incorporates the developments of the past decade. Computing has changed dramatically over that time in ways that have a profound effect on curriculum design and pedagogy. Moreover, the scope of what is called computing has broadened to the point that it is difficult to define it as a single discipline. Past curriculum reports have attempted to merge such disciplines as computer science, computer engineering, and software engineering into a single report about computing education. While such an approach may have seemed reasonable in the past, there is no question that computing in the 21st century encompasses many vital disciplines with their own identity and pedagogical traditions.

Another part of the charter of this group includes supporting the community of professionals responsible for giving a range of courses throughout the United States and beyond; the inclusion of an international perspective was a significant step that introduced greater challenge but was seen as important given the global nature of computing related developments.

1.1 Overall structure of the CC series

Due to the broadening scope of computing—and the feedback received on the initial draft—the CC report was divided into several volumes. This volume focuses specifically on computer engineering (Computing Curricula – Computer Engineering, or CCCE). To encompass the many other disciplines that are part of the overall scope of computing and information technology, however, IEEE-CS and ACM have created additional committees to undertake similar efforts in other areas, including computer science (Computing Curricula - Computer Science, or CCCS, volume published - December 2001), software engineering (Computing Curricula – Software Engineering, or CCSE), and information systems (Computing Curricula – Information Systems, or CCIS).

Once the individual reports have been completed, representatives from all the disciplines will come together to produce an overview volume that links the series together. That overview volume will contain definitions of the various computing disciplines along with
an assessment of the commonalities that exist in the curricular approaches. The structure of the series as a whole is illustrated in Figure 1-1.

This report was published in 2001 by the CC Task Force - Need to change names of documents to take out 2001 from all of them, as well as from text bubbles.

Note: This diagram represents our version of the eventual structure of the CC report. No official organizational endorsements have yet been obtained.

1.2 Overview of the CCCE process

In their charter the main CC Steering Committee wisely gave individual groups freedom to produce a volume that best reflected the needs and requirements of their particular discipline. However they did ask that a certain minimal number of matters should be addressed and consequently that certain components should be included in the report. The minimal set is:

- The body of knowledge (BOK) for the field, i.e., the topics to be covered
- A set of courses that cover the body of knowledge in one or more ways
- The core requirements for the discipline, i.e., the requirements that shall apply to all undergraduates
- The characteristics of graduates of degree programs
This set of requirements was intentionally viewed as being minimalist, as one of the goals of the Steering Committee was to avoid prescription. The experts must have the freedom to act as they see fit. Yet there must be some commonality across the different volumes of the series. It is anticipated that each volume will exceed this minimal set in various ways.

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

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

In order to respond to the challenges of their charter the Computer Engineering Task Force was formed from major Computer Engineering interests from the United States complemented with international involvement. In addition there was some overlap with the original Computer Science Task Force to ensure continuity.

In discharging their duty the Computer Engineering Task Force felt that it was absolutely vital to involve the wider community and indeed several consultative activities took place to confirm the view expressed in this volume. In addition, extensive use was made of the World Wide Web (http://www.eng.auburn.edu/ece/CCCE) to allow any interested party the opportunity to provide comment and suggestion. The published volume has benefited from this wide and vital involvement.

1.2.1 Development of the report
1.2.2 Review of the report

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

1.3 Structure of the CCCE report

The CCCE volume of the report looks specifically at computer engineering. The main body of the report consists of XX chapters. Chapter 2 begins with a survey and analysis of past reports, focusing most closely on Computing Curricula 1991. Chapter 3 outlines the changes that have occurred in computer science since the publication of the CC1991...
report and the implications that those changes have for curriculum design and pedagogy. In Chapter 4, we articulate a set of principles that have guided the development of CC2001 as we attempt to build on the strengths of our predecessors while avoiding some of the problems observed in the earlier reports. Chapters 5 and 6 present overviews of the computer science body of knowledge and the curriculum recommendations that are examined in detail in the appendices. Chapters 7 and 8 describe the courses and approaches we recommend at the introductory and intermediate levels of the curriculum, respectively. Because these courses alone do not constitute a complete undergraduate curriculum, Chapter 9 summarizes additional courses and topics that must be included as part of the academic program. One important aspect of the complete curriculum involves the study of professional practice, which is discussed in Chapter 10. In Chapter 11, we outline a set of characteristics that define the successful computer science graduate. Chapter 12 looks at the problem of teaching computer science and computing-related skills to students in other disciplines. Finally, Chapter 13 offers a variety of strategic and tactical suggestions for implementing the recommendations in this report.

The bulk of the material in the report appears in two appendices. Appendix A looks in detail at the body of knowledge for undergraduate computer engineering. Appendix B consists of full descriptions for the recommended courses that comprise the sample curricula. We hope that providing both the body of knowledge and course descriptions helps departments to create effective curricula more easily than using either of these sources alone.
Chapter 2

Computer Engineering as a Discipline

2.1 Background

Computer engineering embodies the science and technology of design, construction, implementation, and maintenance of software and hardware components of modern computing systems and computer-controlled equipment. Computer engineering has traditionally occupied the territory that lies at the interface between computer science and electrical engineering, evolving over the past three decades as a separate, although intimately related, discipline. Computer engineering is solidly grounded in the theories and principles of computing, mathematics, science, and engineering and applies these theories and principles to solve technical problems through the design of 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 has created an environment in which computers have replaced more conventional electronic devices as system building blocks. This can be seen in the proliferation of mobile telephones, personal digital assistants, location-aware devices, digital cameras, and similar products.

Increasingly, computer engineers are involved in the design of computer-based systems to address highly specialized and specific application needs. Computer engineers work in most industries, including the computer, aerospace, telecommunications, power, manufacturing, defense, and electronics industries. They design high-tech devices ranging from tiny microelectronic integrated-circuit chips, to powerful systems that utilize those chips, to efficient telecommunication systems that interconnect those systems. Applications include consumer electronics (CD players, televisions, VCRs, stereos, microwaves, gaming devices), advanced microprocessors, peripheral equipment, and systems for portable, desktop and client/server computing, communications devices (cellular phones, pagers, personal digital assistants), 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.

Computer engineering continues to be driven by technological advances and innovation. There is now a convergence of television technology, computer technology, and networking technology, resulting in widespread and ready access to information on an
enormous scale. This has created many opportunities and challenges for computer engineering. This convergence of technologies and the associated innovation lie at the heart of economic development and the future of many organizations.

2.2 Evolution of the Field

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

The first computer engineering program to be accredited by ABET (Accreditation Board for Engineering and Technology), in 1971, was at Case Western Reserve University. As of 2002, there were over 140 accredited computer engineering, or similarly named, programs in 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 300 accredited electrical engineering programs.

<table>
<thead>
<tr>
<th>Program Name</th>
<th>1970s</th>
<th>1980s</th>
<th>1990s</th>
<th>2000-02</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Engineering</td>
<td>9</td>
<td>32</td>
<td>44</td>
<td>29</td>
<td>114</td>
</tr>
<tr>
<td>Computer Systems Engineering</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Electrical and Computer Engineering</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>(includes programs previously named EE)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer Science and Engineering</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>other titles</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Totals</td>
<td>14</td>
<td>43</td>
<td>49</td>
<td>36</td>
<td>142</td>
</tr>
</tbody>
</table>

Table 2-1. Summary of ABET-accredited computer engineering programs.

Emergence of software engineering as a separate discipline

2.3 Characteristics of Computer Engineering Graduates

With the ubiquity of computers and computer-based systems in today’s world, 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.
An important distinction should be made between computer engineers, other computer professionals, and engineering technologists. While such distinctions are not always unambiguous, computer engineers generally should satisfy the following three characteristics:

- The ability to design computer systems that include both hardware and software to solve novel engineering problems, subject to trade-offs involving a set of competing goals and constraints. In this context, “design” refers to a level of ability beyond “assembling” or “configuring” systems.
- A breadth of curricular content in mathematics and engineering sciences, beyond that narrowly required for the field, associated with the broader scope of engineering.
- Preparation for professional practice in engineering.

Computer engineers may design and implement computing systems that affect the public. 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. 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 (see chapter 6).

2.3.1 Curricular Preparation

Because of the breadth of the field, curricular content may vary widely among programs, or even among students in the same program. Computer-related coursework typically is drawn from computer organization and architecture, algorithms, programming, databases, networks, software engineering, and communications. Electrical engineering related coursework typically is drawn from circuits, digital logic, microelectronics, signal processing, electromagnetics, and integrated circuit design. Foundational topics typically include basic sciences, mathematics for both discrete and continuous domains, and applications of probability and statistics.

At one extreme, a degree program might provide opportunities for its students to study a wide range of topics spanning the entire field of computer engineering. 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 which they studied, whether it be 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 should be expected of any computer engineering program. The Body of Knowledge, described in Chapter 4, identifies topical areas that may reasonably be expected in all programs, as opposed to those which are often included in some, but not all, programs or those which are generally considered to be elective or specialized topics. From a higher-level perspective, however, there are several characteristics that one can reasonably expect of all computer engineering graduates:

- **System Level Perspective** – Graduates must appreciate the concept of a computer system and the processes involved in constructing or analyzing it. They must have an understanding of its operation that goes to a greater depth than a mere external appreciation of what the system does or the way(s) in which it is used.
- **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 both hardware and software elements, building on prior work, and including at least one major project.
- **Use of Tools** – Graduates should be capable of utilizing a variety of computer-based and laboratory tools for the analysis and design of computer systems, including both hardware and software elements.
- **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.

### 2.3.2 Program and Student Evaluation

Processes for program and student evaluation must accommodate the variation among computer engineering programs. Such evaluation, however, is critical to ensuring that graduates are properly prepared and that programs are evolving to meet the emerging requirements of the field.

Within the United States, ABET accreditation is widely recognized and accepted for this purpose. The current engineering criteria [ABET, 2002] are intended to ensure that all accredited programs satisfy a minimum set of criteria common to all engineering disciplines, criteria specific to each discipline, and any unique objectives that a program has established for itself. A key element of this process is a requirement that each program engage in an ongoing process of self-assessment and continuous improvement. Programs are expected to demonstrate that all graduates at least minimally satisfy a defined set of objectives and outcomes.

In the United Kingdom, benchmarking of degrees has developed in recent years and each institution is required to demonstrate that their degrees meet the requisite benchmark standards for that discipline. One example of these benchmark standards is [UKQAA2000]. This benchmarking defines both threshold (minimal) and modal (average) expectations with respect to demonstrated student knowledge, skills, and judgment. An example of a [threshold/modal] criteria is the following:
• Graduates will be able to produce work involving problem identification, the analysis, the design and the development of a system with appropriate documentation. The work will show [some / a range of] problem solving and evaluation skills drawing on [some/ ] supporting evidence, and demonstrate a [requisite/good] understanding of the need for quality.

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

2.4 Organizational Options

<stand-alone, ECE, CSE departments; inside/outside college of engineering>

2.5 Computer Engineering Outside the United States

Comment: Remove title and write a paragraph that implies implementation of a “core”.

Comment: Two references moved to the end.

Comment: MOVE THIS TO HISTORY SECTION ABOVE – Delete this heading.
Chapter 3
Principles

Computing is a growing and important area of endeavor. The Computer Engineering Task Force established a set of principles to guide their work:

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

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

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

4. **The rapid evolution of computer engineering requires an ongoing review of the corresponding curriculum.** Given the pace of change in the discipline, the professional associations in this discipline must establish an ongoing review process that allows individual components of the curriculum recommendations to be updated on a recurring and timely basis.

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

6. **The Computer Engineering Task Force should seek to identify the fundamental skills and knowledge that all computer engineering graduates must possess.**
7. **The required core of the body of knowledge must be made as small as reasonably possible.** Every effort should be made to keep that core to a minimum to allow flexibility, customization, and choice in other parts of the curriculum so as to enable creation of customized programs.

8. **Computer Engineering must include appropriate and necessary design and laboratory experience components.** A computer engineer requires a design and laboratory experience that should provide design, problem solving, and debugging experience.

9. **The Computer Engineering core must acknowledge that engineering curricula are often subject to accreditation, licensure, or governmental constraints.** The Computer Engineering report should not just fit within existing constraints, but instead should provide guidance for their evolution.

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

11. **The Computer Engineering report must include discussions of strategies and tactics for implementation along with high-level recommendations.** Although it is important for Computing Curricula to articulate a broad vision of computing education, the success of any curriculum depends heavily on implementation details. **Sample curricula will be provided.**

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

13. **The Computer Engineering final report must strive to be international in scope.** Despite the fact that curricular requirements differ from country to country, Computing Curricula is intended to be useful to computing educators throughout the world.
Chapter 4

Overview of the Computer Engineering Body of Knowledge

4.1 Introduction

In developing a curriculum for undergraduate study in computer engineering, one of the first steps is to identify and organize subject material that would be appropriate for that level. The Computer Engineering Task Force sought to accomplish this goal by first defining the primary disciplines that make up the body of knowledge for computer engineering. Some of these areas were determined to contain material that should be studied in all computer engineering programs, while other areas contain material that might, or might not, be included, depending on the specific educational objectives of that program. The areas so identified are as follows.

**CCCE Discipline Areas Containing Core Material**

- CE-ALG Algorithms and Complexity
- CE-CAO Computer Architecture and Organization
- CE-CSE Computer Systems Engineering
- CE-CSY Circuits and Systems
- CE-DIG Digital Logic
- CE-DSC Discrete Structures
- CE-DSP Digital Signal Processing
- CE-DSY Distributed Systems
- CE-ELE Electronics
- CE-ESY Embedded Systems
- CE-HCI Human-Computer Interaction
- CE-INM Information Management
- CE-INS Intelligent Systems
- CE-NWK Computer Networks
- CE-OPS Operating Systems
- CE-PRF Programming Fundamentals
- CE-PRS Probability & Statistics
- CE-SPR Social and Professional Issues
- CE-SWE Software Engineering
- CE-TAV Test and Verification
- CE-VLS VLSI/ASIC Design

After defining the above areas, each Task Force member was assigned the responsibility of developing initial drafts defining the body of knowledge for one or more areas. In some cases, new members were added to the Task Force to cover areas of expertise outside of those represented on the Task Force. Then, a second task force member was assigned to critique and revise each initial draft. After revision, the draft was sent to the entire task force for comment. At the completion of this process, the entire Task Force met as a group to review the draft body of knowledge, with follow-up
modifications made as appropriate. It was released for public review at this time, with reviews solicited at a number of meetings, conferences and other sources. An NSF sponsored workshop was held in conjunction with FIE’2002. Reviewers from academia and industry were invited to provide comments on the preliminary version of the Body of Knowledge. The Body of Knowledge was also presented and discussed at a variety of conferences through panel discussions and poster sessions [refs for FIE02 panel, ASEE02, John’s poster in 03, etc.] Finally, the Body of Knowledge and the remainder of the document were made available on the task force web site for review by the entire computer engineering community.

4.2 Structure of the body of knowledge

The body of knowledge is organized hierarchically into three levels:

- The highest level of the hierarchy is the area, which represents a particular disciplinary sub-field. Each area is identified by a three-letter abbreviation, such as DLG for Digital Logic or CAO for Computer Architecture and Organization.
- Each area is broken down into smaller divisions called units, which represent individual thematic modules within that area. Each unit is identified by adding a numeric suffix to the area name; as an example, CAO3 is a unit on Memory System Organization and Architecture.
- Each unit is further subdivided into a set of topics, which are the lowest level of the hierarchy. Associated with the units are learning outcomes which identify the related technical skills that need to be addressed; see section 4.2.1 below.

There is a certain duality between the knowledge elements and the related learning objectives or learning outcomes. Different people will place different levels of emphasis on each. In this document the view is that they are complementary.

4.3 Learning Outcomes

In order to capture the provision of all the various skills that are to be addressed in a unit of learning the concept of learning outcomes is utilized. The emphasis on learning is important. The concept of learning outcomes is a mechanism for describing, not just knowledge and relevant practical skills, but also personal and transferable skills. They can be associated with a knowledge unit or a class or even a degree program and can be used to convey a number of aspects of the ethos of a course.

Any specification of a course will include both knowledge and associated learning outcomes. In designing courses some designers start with knowledge, others start with the learning outcomes. In reality a combination of the two approaches is often seen as the most appropriate.

Since learning outcomes imply assessment and assessment guides learning, considerable care is required in selecting and formulating these. Excessive numbers of very detailed learning outcomes can lead to bureaucracy and tedium (highly
The existence of these must not inhibit course development but should enhance it.

Learning outcomes will be associated with the knowledge units and with modules, with the latter constituting the formal units of assessment. The number of learning outcomes per knowledge unit or module should be a small number (e.g. at most four or five). The learning outcomes for a module will naturally build on the knowledge units and the associated practical skills. They will tend to be of the form:

\[
\text{demonstrate the acquisition of competence (i.e. the ability to apply knowledge and practical skills to solve a problem).}
\]

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

4.4 Core and Elective Units

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

To implement this principle, the Computer Engineering Task Force has defined a minimal core comprising those units for which there is broad consensus that the corresponding material is essential to anyone obtaining an undergraduate degree in computer engineering. Units that are taught as part of an undergraduate program, but which fall outside the core, are considered to be elective. Some institutions might deem that these elective areas should be required for their program.

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

- **The core is not a complete curriculum.** The core is intended to be minimal and it does not, by itself, constitute a complete undergraduate curriculum. Every undergraduate program must include additional elective units from the body of knowledge, although the Computer Engineering report does not define what those units will be. A complete curriculum will also contain supporting areas that are covered in the traditional mathematics, natural sciences, humanities, and social sciences. These will be discussed in Chapter 7. Again, some of these supporting courses may be required for individual programs, but are not detailed here.

- **Core units are not necessarily limited to a set of introductory courses taken early in the undergraduate curriculum.** Although many of the units defined as core are indeed introductory, there are also some core units that clearly must be covered...
only after students have developed significant background in the field. For example, the Task Force believes that all students must develop a significant application at some point during their undergraduate program. The material that is essential to successful management of projects at this scale is therefore part of the core, since it is required of all students. At the same time, the project course experience is very likely to come toward the end of a student's undergraduate program. Similarly, introductory courses may include elective units alongside the coverage of core material. The designation core simply means required and says nothing about the level of the course in which it appears.

4.5 Units and Time Required for Coverage

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

- The Task Force does not seek to endorse the lecture format. Even though a metric with its roots in a classical lecture-oriented form has been used, the Task Force believes there are other styles - particularly given recent improvements in educational technology - that can be at least as effective. For some of these styles, the notion of hours may be difficult to apply. Even so, the time specifications should at least serve as a comparative measure, in the sense that a 5-hour unit will presumably take roughly five times as much time to cover as a 1-hour unit, independent of the teaching style.

- The hours specified do not include time spent outside of a class. The time assigned to a unit does not include the instructor's preparation time or the time students spend outside of class. As a general guideline, the amount of out-of-class work is approximately three times the in-class time. Thus, a unit that is listed as requiring 3 hours will typically entail a total of 12 hours (3 in class and 9 outside) of student effort.

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

4.6 Core Hours and a Complete Program
A summary of the body of knowledge - showing the areas, units, which units are core, and the minimum time required for each - appears as Figure 4-1. The details of the body of knowledge appear in Appendix A.

The core hours as specified in Figure 4-1 total 420 hours. Recall that an hour refers to contact hours and not credit hours. Assuming a typical 15 week semester, a typical three credit hour course would have about 42 contact hours for presentation of material. The 466 core hours are thus roughly equivalent to ten 3-credit hour courses. This number is approximately one quarter of the 128 credit hours specified in a typical engineering program.

This leaves ample room for the addition of laboratory courses, a capstone project, and electives that allow an institution to customize their program. For example, ABET currently requires one and one-half years of engineering topics, along with one year of math and basic science. Counting the Discrete Structures area and the Probability and Statistics areas as mathematics, the core hours listed in Table 4.1 would constitute approximately two-thirds of the required minimum engineering content.

Comment: VPN – 420 was decided upon as our target. The actual hours will be the result of our Core Hours Allocation Exercise.

Comment: VPN – This still needs some work.
<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code</td>
<td>Course Title</td>
<td>Credits</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>CE-DP9</td>
<td>Poles and zeros in the z plane.</td>
<td></td>
</tr>
<tr>
<td>CE-DP10</td>
<td>IIR filters, frequency and phase response.</td>
<td></td>
</tr>
<tr>
<td>CE-DP11</td>
<td>Design of IIR Filters.</td>
<td></td>
</tr>
<tr>
<td>CE-ELE0</td>
<td>History and overview of electronics</td>
<td>(1)</td>
</tr>
<tr>
<td>CE-ELE1</td>
<td>Electronic properties of materials</td>
<td>(4)</td>
</tr>
<tr>
<td>CE-ELE2</td>
<td>Diodes and diode circuits</td>
<td>(5)</td>
</tr>
<tr>
<td>CE-ELE3</td>
<td>MOS transistors and biasing</td>
<td>(3)</td>
</tr>
<tr>
<td>CE-ELE4</td>
<td>MOS logic families</td>
<td>(9)</td>
</tr>
<tr>
<td>CE-ELE5</td>
<td>Bipolar transistors and logic families</td>
<td>(5)</td>
</tr>
<tr>
<td>CE-ELE6</td>
<td>Design parameters and issues</td>
<td>(5)</td>
</tr>
<tr>
<td>CE-ELE7</td>
<td>Storage elements</td>
<td>(3)</td>
</tr>
<tr>
<td>CE-ELE8</td>
<td>Interfacing logic families and standard buses</td>
<td>(3)</td>
</tr>
<tr>
<td>CE-ELE9</td>
<td>Operational amplifiers</td>
<td>(6)</td>
</tr>
<tr>
<td>CE-ELE10</td>
<td>Data conversion circuits</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ELE11</td>
<td>SPICE circuit simulation</td>
<td>(3)</td>
</tr>
<tr>
<td>CE-ELE12</td>
<td>Electronic voltage and current sources</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ELE13</td>
<td>Linear amplification and biasing</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ELE14</td>
<td>Single-transistor amplifiers</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ELE15</td>
<td>Multistage transistor amplifiers</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ELE16</td>
<td>Power circuits</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ELE17</td>
<td>Feedback in electronics</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ELE18</td>
<td>Active filters</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ELE19</td>
<td>Integrated circuit building blocks</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ELE20</td>
<td>Circuits for wireless applications</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-HC1</td>
<td>HCI in computer engineering</td>
<td>[2]</td>
</tr>
<tr>
<td>CE-HC3</td>
<td>Building a simple graphical user interface</td>
<td>[4]</td>
</tr>
<tr>
<td>CE-HC4</td>
<td>Human-centered software evaluation</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-HC5</td>
<td>Human-centered software development</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-HC6</td>
<td>Interactive graphical user-interface design</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-HC7</td>
<td>Graphical user-interface programming</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-HC8</td>
<td>Graphics and visualization</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-HC9</td>
<td>HCI aspects of multimedia systems</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INS1</td>
<td>Intelligent systems in computer engineering</td>
<td>[2]</td>
</tr>
<tr>
<td>CE-INS2</td>
<td>Overview of technologies</td>
<td>[1]</td>
</tr>
<tr>
<td>CE-INS3</td>
<td>Fundamental issues in intelligent systems</td>
<td>[core]</td>
</tr>
<tr>
<td>CE-INS4</td>
<td>Search and constraint satisfaction</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INS5</td>
<td>Knowledge representation and reasoning</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INS6</td>
<td>Agents</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INS7</td>
<td>Natural language processing</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INS8</td>
<td>Machine learning and neural networks</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INS9</td>
<td>AI planning systems</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INS10</td>
<td>Robotics</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-OPS0</td>
<td>History and overview of operating systems</td>
<td>(1)</td>
</tr>
<tr>
<td>CE-OPS1</td>
<td>Operating system function and design</td>
<td>(2)</td>
</tr>
<tr>
<td>CE-ESY0</td>
<td>Introduction to embedded systems</td>
<td>(1)</td>
</tr>
<tr>
<td>CE-ESY1</td>
<td>Embedded microcontrollers</td>
<td>(6)</td>
</tr>
<tr>
<td>CE-ESY2</td>
<td>Embedded programs</td>
<td>(3)</td>
</tr>
<tr>
<td>CE-ESY3</td>
<td>Real-time operating systems</td>
<td>(3)</td>
</tr>
<tr>
<td>CE-ESY4</td>
<td>Low-power computing</td>
<td>(3)</td>
</tr>
<tr>
<td>CE-ESY5</td>
<td>Reliable system design</td>
<td>(3)</td>
</tr>
<tr>
<td>CE-ESY6</td>
<td>Design methodologies</td>
<td>(3)</td>
</tr>
<tr>
<td>CE-ESY7</td>
<td>Tool support</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ESY8</td>
<td>Embedded multiprocessors</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ESY9</td>
<td>Networked embedded systems</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ESY10</td>
<td>Interfacing and mixed-signal systems</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-ESY11</td>
<td>Embedded computing system examples</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-INM1</td>
<td>Information management in computer engineering</td>
<td>[2]</td>
</tr>
<tr>
<td>CE-INM2</td>
<td>Information systems</td>
<td>[3]</td>
</tr>
<tr>
<td>CE-INM3</td>
<td>Database systems</td>
<td>[4]</td>
</tr>
<tr>
<td>CE-INM4</td>
<td>Data modeling</td>
<td>[4]</td>
</tr>
<tr>
<td>CE-INM5</td>
<td>Relational databases</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INM6</td>
<td>Database query languages</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INM7</td>
<td>Relational database design</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INM8</td>
<td>Transaction processing</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INM9</td>
<td>Distributed databases</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INM10</td>
<td>Physical database design</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-INM11</td>
<td>Multimedia information and systems</td>
<td>[elective]</td>
</tr>
<tr>
<td>CE-NWK0</td>
<td>History and overview of networks</td>
<td>(1)</td>
</tr>
<tr>
<td>CE-NWK1</td>
<td>Communications network architecture</td>
<td>(4)</td>
</tr>
<tr>
<td>CE-NWK2</td>
<td>Communications network protocols</td>
<td>(6)</td>
</tr>
<tr>
<td>CE-NWK3</td>
<td>Local and wide area networks</td>
<td>(6)</td>
</tr>
<tr>
<td>CE-NWK4</td>
<td>The web as an example of client-server computing</td>
<td>(6)</td>
</tr>
<tr>
<td>CE-NWK5</td>
<td>Data security and integrity</td>
<td>(6)</td>
</tr>
<tr>
<td>CE-NWK6</td>
<td>Performance evaluation</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-NWK7</td>
<td>Data communications</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-NWK8</td>
<td>Wireless and mobile computing</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-NWK9</td>
<td>Network management</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-NWK10</td>
<td>Compression and decompression</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-NWK11</td>
<td>Multimedia data technologies</td>
<td>(elective)</td>
</tr>
<tr>
<td>CE-PRF0</td>
<td>History and overview of program fundamentals</td>
<td>(1)</td>
</tr>
<tr>
<td>CE-OPS2. Operating system principles (3)</td>
<td>CE-PRF1. Fundamental programming constructs (7)</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CE-OPS4. Scheduling and dispatch (3)</td>
<td>CE-PRF3. Fundamental data structures (17)</td>
<td></td>
</tr>
<tr>
<td>CE-OPS6. Device management (elective)</td>
<td>CE-PRF5. Recursion (6)</td>
<td></td>
</tr>
<tr>
<td>CE-OPS7. Security and protection (elective)</td>
<td>CE-PRF6. Object-oriented programming (elective)</td>
<td></td>
</tr>
</tbody>
</table>

| CE-PRF1. Fundamental programming constructs (7) |
| CE-PRF2. Algorithms and problem-solving (9)    |
| CE-PRF3. Fundamental data structures (17)      |
| CE-PRF4. Programming Paradigms (8)             |
| CE-PRF5. Recursion (6)                        |
| CE-PRF6. Object-oriented programming (elective) |
| CE-PRF7. Event-driven and concurrent programming (elective) |
| CE-PRF8. Using APIs (elective)               |

| CE-PRF1. Fundamental programming constructs (7) |
| CE-PRF2. Algorithms and problem-solving (9)    |
| CE-PRF3. Fundamental data structures (17)      |
| CE-PRF4. Programming Paradigms (8)             |
| CE-PRF5. Recursion (6)                        |
| CE-PRF6. Object-oriented programming (elective) |
| CE-PRF7. Event-driven and concurrent programming (elective) |
| CE-PRF8. Using APIs (elective)               |

| CE-PS1. Discrete probability (8)               |
| CE-PS2. Continuous probability (8)             |
| CE-PS3. Sampling distributions (4)             |
| CE-PS4. Stochastic Processes (8)               |
| CE-PS5. Sampling distributions (4)             |
| CE-PS6. Estimation (4)                        |
| CE-PS7. Hypothesis tests (2)                   |
| CE-PS8. Correlation and regression (2)         |

| CE-PS1. Discrete probability (8)               |
| CE-PS2. Continuous probability (8)             |
| CE-PS3. Sampling distributions (4)             |
| CE-PS4. Stochastic Processes (8)               |
| CE-PS5. Sampling distributions (4)             |
| CE-PS6. Estimation (4)                        |
| CE-PS7. Hypothesis tests (2)                   |
| CE-PS8. Correlation and regression (2)         |

| CE-SPR0. History and overview of social and professional issues (1) |
| CE-SPR1. Social context of computing (3)                    |
| CE-SPR2. Methods and tools of analysis (2)                  |
| CE-SPR3. Professional and ethical responsibilities (3)      |
| CE-SPR4. Risks and liabilities of computer-based systems (2)|
| CE-SPR5. Intellectual property (3)                          |
| CE-SPR6. Privacy and civil liberties (2)                    |
| CE-SPR7. Computer crime (elective)                          |
| CE-SPR8. Economic issues in computing (elective)             |
| CE-SPR9. Philosophical frameworks (elective)                 |

| CE-SWE0. History and overview of software engineering (1)    |
| CE-SWE1. Software processes (2)                              |
| CE-SWE2. Software requirements and specifications (2)        |
| CE-SWE3. Software design (2)                                 |
| CE-SWE4. Software testing and validation (2)                 |
| CE-SWE5. Software evolution (elective)                       |
| CE-SWE6. Software tools and environments (2)                 |
| CE-SWE7. Software project management (elective)              |

| CE-SWE0. History and overview of software engineering (1)    |
| CE-SWE1. Software processes (2)                              |
| CE-SWE2. Software requirements and specifications (2)        |
| CE-SWE3. Software design (2)                                 |
| CE-SWE4. Software testing and validation (2)                 |
| CE-SWE5. Software evolution (elective)                       |
| CE-SWE6. Software tools and environments (2)                 |
| CE-SWE7. Software project management (elective)              |

| CE-VLS1. MOS Transistor Fundamentals                        |
| CE-VLS2. Processing and Layout                             |
| CE-VLS3. Function of the Basic Inverter Structure           |
| CE-VLS4. Circuit Characterization and Performance           |
| CE-VLS5. Combinational Logic Circuits                      |
| CE-VLS6. Sequential Logic Circuits                         |
| CE-VLS7. Alternative Circuit Structures/Low Power Design   |
| CE-VLS8. Semiconductor Memories and Array Structures       |
| CE-VLS9. Chip Input/Output Circuits                        |
| CE-VLS10. Semi custom Design Technologies                  |
| CE-VLS11. ASIC Design Methodology                          |

| CE-VLS1. MOS Transistor Fundamentals                        |
| CE-VLS2. Processing and Layout                             |
| CE-VLS3. Function of the Basic Inverter Structure           |
| CE-VLS4. Circuit Characterization and Performance           |
| CE-VLS5. Combinational Logic Circuits                      |
| CE-VLS6. Sequential Logic Circuits                         |
| CE-VLS7. Alternative Circuit Structures/Low Power Design   |
| CE-VLS8. Semiconductor Memories and Array Structures       |
| CE-VLS9. Chip Input/Output Circuits                        |
| CE-VLS10. Semi custom Design Technologies                  |
| CE-VLS11. ASIC Design Methodology                          |
Chapter 5

Integration of Engineering Practice into the Computer Engineering Curriculum

By its very nature, any curriculum in Computer Engineering must exhibit an engineering ethos. It is desirable that this ethos should permeate all years of the curriculum and do so in a consistent manner. Such an approach has the effect of introducing students to engineering (and in particular computer engineering), teaching them to think and function as engineers, and setting expectations for the future.

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

5.1 The Nature of Computer Engineering

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

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

5.2 Design in the Curriculum

Engineering depends heavily on design, a creative ability that is required for the development of better devices, systems, processes, and new products. In many cases new designs are prompted by: seeking to exploit new developments in related technologies; seeking to improve on existing products (e.g. making them less expensive, safer, more flexible, lighter in weight); and identifying deficiencies or weaknesses in existing products. But novel ideas are especially important.
Design is fundamental to all engineering. For the computer engineer, design relates to software and hardware components of modern computing systems and computer-controlled equipment. Computer engineers apply the theories and principles of science and mathematics to design hardware, software, networks, and processes and to solve technical problems. Continuing advances in computers and digital systems have created opportunities for professionals capable of applying these developments to a broad range of applications in engineering. Fundamentally it is about making well considered choices and 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; fundamentally these tend to be related to simplicity and elegance.

5.2.1 Design Throughout the Curriculum

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 has to occur will provide a framework within which choices have ultimately to be made. This framework can be influenced by the skills of individuals, by such matters as reliability requirements or security considerations, and by the environment in which a device has to be used (e.g. extremes of weather or temperature, by novice users).

An area of particular concern to the computer engineer is the software / hardware boundary where difficult trade-off decisions provide 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 about hardware/software trade-offs can occur and this leads naturally to the design of special purpose computers and systems.

At a different level there are all the difficult issues of software design. Particular aspects of this relate to the human-computer interface. Addressing this comprehensively can lead to considerations about multi-media, graphics, animation and a whole host of technologies.

In short, design is central to computer engineering and the challenges can be immense.

5.2.2 The Culminating Design Experience

The concept of a culminating design project is widely valued as an important experience that occurs at the end of a curriculum. Students take a significant problem associated with a discipline and in solving the problem they have the opportunity to demonstrate their ability to provide a solution. Typically the solution must involve the design and implementation of some product containing hardware and/or software components. Ideally this design experience should incorporate engineering standards and realistic constraints.
Add a statement about teamwork, etc – but not just ABET

The benefits stemming from this experience include:

- demonstration of the ability to integrate concepts from several different subjects in coming to a solution
- demonstration of the application of disciplines associated with Computer Engineering
- production of a well-written document detailing the design and the design experience
- demonstration of creativity and innovation
- development of time management and planning skills
- self-awareness opportunities provided by assessment of achievement as part of the final report

Depending on the approach to assessment other opportunities arise. Assessment may include a demonstration, a presentation, an oral examination, production of a web page, industry review, and many other possibilities.

Although not listed in the core body of knowledge, the culminating design experience must be an integral part of the undergraduate experience.

5.3 The Laboratory Experience

As in any engineering curriculum, it is important that computer engineering students have ample opportunities to observe and explore characteristics and behaviors of actual devices, systems, and processes. This includes designing, implementing, testing, and documenting hardware and software, designing experiments to acquire data, analyzing and interpreting that data, and in some cases using that data to correct or improve the design. This is usually done most effectively within a laboratory setting, either as an integral part of a course or in a separate stand-alone course.

5.3.1 Laboratory Experiments

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

Laboratories should include some physical implementation of designs, such as construction of electronic circuits, digital systems, computer systems and/or software

Comment: VPN – Should we explicitly mention breadboarding, microprocessor interfacing, prototyping with FPGAs, etc.?
systems. However, 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 physically design and implement, or where it might be difficult to acquire the detailed information necessary to study their behavior. Of particular importance is for students to explore the effects of design tradeoffs.

Students should learn to record laboratory activity to document and keep track of all design activities, conducted experiments and their measured/observed results, both good and bad. Keeping records of component values, design decisions, etc. is valuable to the overall project.

The laboratory should also assist students in learning more practical issues:

- safety in all laboratories, but with especially where electronics and electricity pose dangers
- proper use of computers and other test equipment
- treating laboratories as places of serious study and endeavor (like a library)

5.3.2 Practical activity

Observations about the role of engineering practice in the computer engineering curriculum have implications for the nature of the practical activities that students undertake throughout their education. A careful approach to assignment selection and assessment can generate within the student a sense of excitement as well as a sense of accomplishment leading to increased self-confidence and even self-generated ambitions which are the seed of innovation and creativity.

At the formative stages of their education a more basic set of concerns needs to be addressed. This gives rise to the following important elements of practical activity:

- seeing high quality practical work as challenging with assessment reflecting rewards for innovation
- seeing tool identification and deployment as important
- building electronic circuits and devices
- understanding of the processes and concerns associated with product development and manufacture
- recognizing the possibilities of trade-offs and being able to resolve decisions in this area; the hardware / software trade-off is of particular concern
- recognizing the potential of hardware and software and the benefits and drawbacks of each.

Fundamentally, carefully planned and carefully organized practical assignments should help students develop the confidence and the know-how to build new devices and to appreciate the support that can be mobilized to realize ambition, e.g. the use of technical staff and workshop teams, or professionals from other disciplines.
5.4 The Role of Engineering Tools

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

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

Not every computer engineering program will incorporate all of these tools. Appropriate tools should be incorporated throughout the program of study, consistent with the program’s goals and objectives.

Identifying the scope for the development of tools and components generally is yet another role for the computer engineer. A natural subsequent activity is engaging in the design and development of these. Such activities need to be guided by concerns for quality in all its different guises – safety, usability, reliability, and so on.

5.5 Applications of Computer Engineering Principles

Given the nature of computer engineering and the expectations of students entering such courses, applications have a fundamental role to play in terms of:

- motivating students in their studies
- guiding their thinking and ambition
- providing justification for the inclusion and the prominence of certain material
- demonstrating the application of theoretical ideas

Comment: We deleted “and re-use” from this section, since we felt that issues of re-use, IP, etc. should go into a different section than “tools”.

Comment: VPN – There was originally something here on “re-use”: Perhaps this should go into the design section above?

Comment: VPN – More descriptive term than “CPE Principles”? Case studies are the focus here, but is that too narrow?

Deleted: The deployment of tools can be viewed as a form of re-use. The use of libraries of components (software as well as hardware) is a more obvious form of re-use and again this is fundamental to facilitating the whole engineering process.

Deleted: In many cases databases can be used to hold libraries of information about components, (for instance, part numbers, footprints for layout, electrical information, symbols for schematic capture).

Deleted: In addition certain tools are relevant to both hardware and software and systems involving combinations of these – thus a range of simulation and modelling tools, configuration management and version control tools, and so on.

Para:

Comment: VPN – More descriptive term than “CPE Principles”? Case studies are the focus here, but is that too narrow?
This can be achieved through a whole range of possible routes: the use of up-to-date and topical case studies, guided reading, assessments, speakers from industry, and so on. This can also happen at a whole range of levels, for instance, chip design, software tools, and entire systems. Suitable applications can also provide a forum for group work, perhaps of an interdisciplinary nature. To this end, all computer engineering students should engage in an in-depth study of some significant application that uses computing engineering in a substantive way.

Computer engineering students will typically have a wide range of interests and professional goals. For many students, in depth study of some aspect of computer engineering will be extremely useful. Such work might be accomplished 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. Activities of this kind can be interdisciplinary in nature and so provide opportunities for particularly beneficial kinds of group activity. Thus the computer engineer may have to work with professionals from other disciplines and these can be seen to include computer scientists, electronics engineers, psychologists, economists, financial experts, marketing personnel, product designers, and so on.

5.6 Complementary Skills

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

One aspect of this is to ensure that students possess a set of transferable or personal skills such as communication skills, group working skills, presentational skills, etc. But additionally one could include: library and research skills as well as professional skills that can include time management, project management, information management, understanding industry, career development skills, self-awareness, keeping up-to-date and so on. Some see these skills as having to be woven into the fabric of the traditional curriculum. Within these various skills are elementary aspects and more advanced aspects and these ought to be addressed at different and relevant stages of the curriculum. From a motivational perspective these skills need to be addressed in the context of computer engineering and in a manner that highlights their relevance and importance to the discipline.

There is always a danger that time spent on complementary skills can absorb excessive amounts of time and effort and swamp or displace the more traditional material, thereby reducing knowledge. There are delicate issues of balance here, and typically a subtle approach to both teaching and assessment is required to ensure that there is not imbalance in the curriculum. When this happens fun and excitement are often sacrificed for more bureaucratic concerns, and that is both unfortunate and undesirable.
5.7 Communication Skills

A widely-held theme among employers is that 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 having to pass requirements to a workshop or to technical support staff – which in an industrial setting may be local or remote. Providing clear and succinct instructions and having proper regard for the role and purpose of support staff, affects the efficiency and the nature of the working environment. It is a fundamental communication skill.

Taking these considerations into account, in their studies students should learn to:

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

While institutions may adopt different strategies to accomplish these goals, the program of each computer engineering student must include numerous occasions for improving these skills in a way that emphasizes writing, speaking and active listening skills. There also needs to be recognition of elementary aspects as well as progression in terms of enhancing these skills (not by accident but through conscious strategy) and opportunities for demonstrating the acquisition of advanced skill levels.

At a minimum, a computer engineering curriculum should require:

• course work that emphasizes the mechanics and process of writing
• 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. Communication skills should not be seen as separate, but should instead be fully incorporated into the computer engineering curriculum and its requirements.
A complementary and important set of communication skills arise in the context of electronic media. Increasingly these have a central role to play in the life of the engineer. Apart from the obvious need to address e-mail, web design, etc. to some level, ideas on effective cooperative working and group learning have an increasingly prominent place in the curriculum.

5.8 Teamwork Skills

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

Many of the problems of teamwork relate to communication skills. Where multidisciplinary teams are involved roles tend to be determined, at least in part, by the technical expertise of individuals. But generally in team activity, there are important additional issues related to such matters as the nature and composition of teams, roles within teams, organization of team meetings, methods of reaching consensus and for recording decisions, the importance of interfaces, the nature of deadlines and planning, the importance and the nature of quality control mechanisms, and so on.

To ensure that students have the opportunity to acquire these skills as undergraduates, computer engineering programs should include the following:

- opportunities to work in teams beginning relatively early in the curriculum.
- a significant project that involves a complex implementation task in which both design and implementation are undertaken possibly also by a small student team.

5.9 Lifelong Learning Skills

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

A number of basic strategies can be identified. First, the curriculum itself must be up-to-date, the equipment has to be up-to-date, and faculty need to be engaged in relevant scholarship. Relevant reference material (textbooks, software, web sites, case studies, illustrations, etc.) can be referenced with the aim of identifying sources of up-to-date and interesting information. But in addition there are more fundamental considerations.
Lifelong learning is essentially an attitude of mind. That can be fostered by approaches to teaching and learning that continually question and challenge, highlighting opportunities for advances. Students can be challenged by assessments and exercises that seek to explore new avenues. It is also essential to see learning as an aspect that merits attention throughout the curriculum and this being planned. For instance the following stages have been identified in [Fellow 2002]:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Student</th>
<th>Instructor</th>
<th>Instructional Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dependent</td>
<td>authority/coach</td>
<td>lecture, coaching</td>
</tr>
<tr>
<td>2</td>
<td>interested</td>
<td>motivator/guide</td>
<td>inspirational lecture, discussion group</td>
</tr>
<tr>
<td>3</td>
<td>involved</td>
<td>facilitator</td>
<td>discussion lead by instructor who participates as equal</td>
</tr>
<tr>
<td>4</td>
<td>self-directed</td>
<td>consultant</td>
<td>internships, dissertation, self-directed study group</td>
</tr>
</tbody>
</table>

5.10 **The Business Perspective**

The computer engineer needs to have an understanding of the processes that lead to the development of new products. There are various facets to this. Fundamentally the computer engineer needs to develop an appreciation of creativity and innovation and have an eye to new opportunities for wealth creation.

To complement the technical side computer engineers need to have an appreciation of the business perspective associated with the development of new products and devices. In support of this there is benefit in:

- understanding the importance of the financial and economic imperatives associated with new products and even new organizations
- appreciating the relevance of the marketing perspective
- knowing what is involved in product design and product acceptability
- appreciating the benefits of team work, often multi-disciplinary in nature

5.10 **The Elements of an Engineering Ethos**

Summing up, the characteristics of an engineering education include the following:

- seeing their discipline being based on sound principles and sound underpinnings; it is important 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 tools; being able to respond to the challenges of building them and recognizing the need to use these properly and effectively
• recognizing the range of applications for their work
• seeing innovation and creativity as important and understanding relevant business perspectives and opportunities
• recognizing the importance of team activity and the strengths that can be derived from this
• understanding principles of product design including health and safety as well as marketing issues
• seeing disciplined approaches as being important
• recognizing the importance of understanding the relevant professional, ethical and legal issues and the framework within which engineers needs to operate
• being able to address a significant problem in computer engineering, and demonstrating the ability to deploy an appropriate selection of tools and techniques as well as a disciplined approach in arriving at a solution of the problem.

References

[Fellows 2002]

[Feiel et al, 2002]

Deleted:

As mentioned earlier the entire range of considerations that contribute to an engineering ethos includes attention to professional, legal and ethical issues as well as an ability to engage in an individual capstone project. Beyond these, this chapter has sought to address the range of basic ingredients that must be assembled and carefully integrated in order to ensure that graduates are truly initiated in the best traditions of engineering. The next chapter seeks to address this difficult task of course design.
Chapter 6

Professionalism

6.1 Introduction

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

Applying ethics in technology is an indispensable component of a developing technology that has support in a sustainable manner. The social context of engineering should be an integral component of engineering design and development. It would not be expected that the design and construction of a building, bridge, or tunnel would be void of social context. Likewise, it would not be expected that the design and construction of a computer system such as an x-ray machine would be void of that same context. Computer engineers should apply best practices to their work. They should be expected to follow prescribed rules of professional practice and not engage in extraneous activities that would tarnish their image or that of their practicing colleagues.

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

6.2 Decisions in a Societal Context

As designers of computing systems, computer engineers will face many decisions in their careers. Many of these decisions will be technical ones. However, many of them will involve a societal context. Computer engineers should understand the legal ramifications
of contract law, business organization and management, and corporate law. Graduate engineers often assume managerial positions early in their career and need to be prepared to make decisions within a legal context. They also need to understand the customs and practices that govern business.

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

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

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

Best practices begin in the instructional laboratory. Educational institutions should encourage behavioral patterns in laboratories that reflect best practices. Such patterns set a level or norm of behavior and elevate the professional expectations of students. They also create a learning environment that is supportive of the professional tenets to which computer engineers aspire. For example, consider safety issues. Institutions should establish guidelines on 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 used to help students understand the environments in which they will have to function.

What should computer engineers do when facing moral and ethical dilemmas? Obviously, the answer is not always clear. It depends on the situation—what it is, where it takes place, and the social context of the situation. Faced with contrasting ethical philosophies coupled with external forces of economics, politics, personal well-being, and human morality, how might a computer engineer be guided on doing the right thing?

6.3 Fostering Professional Practice

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

Computer engineers can use the codes from several societies to guide them. Some of these codes include:

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

Although each of these codes focus on the particular purposes of the society or societies sponsoring them, common themes pervade all of them. Fundamental to all these codes are the responsibilities of the computing and engineering professional to the public and to the public good. Additionally, these codes address issues of conflicts of interest, scope of competence, objectiveness and truthfulness, deception, and professional conduct.
The precepts delineated within these codes should be the hallmark of all practicing computer engineers. Computer engineers should adopt the tenets of these codes of ethics and professional practices in all the work they do. It is incumbent upon educational programs to educate computer engineers to embrace these tenets for the benefit of their own careers and for the benefit of the computing and engineering professions.

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

6.4 Summary

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

Additionally, computer engineers should understand that the technology they design affects not only a small group of people or a nation. It affects all of society. For example, a company could design a product in one nation. A second nation may develop the product, while a third nation could manufacture it, and different peoples from many nations could use it. Computer engineers may be involved in all aspects of the product---from its design to its delivery. Therefore, computer engineers should be sensitive to the customs and laws affecting those people involved in the entire process.

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

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

Chapter 7

Curriculum Implementation Issues

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

7.1 General Considerations

There is a great variety of knowledge, practical skills, transferable skills and attitudes that need to be addressed within the one single framework. A program should exhibit an obvious and consistent ethos that permeates a complete program of study. Then 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 relatively settled material (e.g., circuits or supporting mathematics courses) on the one hand, versus more recent material on the other hand, among four years of study. Computer engineering is a discipline in which the rate of change is very swift and this is likely to remain so. Traditional approaches to course design suggest that fundamental and core material should appear at the start of a program. The logic of this is that, by its very nature, this material ought to exhibit a level of permanence and durability and so should be unlikely to change certainly over the lifetime of the program. Then students can build on these foundations as they move forward to the later parts of the program.

This view needs to be tempered by consideration of the students’ point of view. Students come to university to study computer engineering. If, during their initial experiences, they are presented only with courses on mathematics, science, etc. coupled with a dry and arid approach to transferable skills, then that can create frustration and disillusionment in the student. This must be avoided. So the traditional view needs to be tempered by considerations of the student perspective.

It is desirable to plan that classes in very new topics are positioned in the later years. These new topics are then at the forefront of research and development and students can genuinely claim to be up-to-date in their subject area. That is important; 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

39
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 possible degree programs are perfectly possible 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
- real-time systems

Another consideration is how many modules can be designed specifically for computer engineering students and how many will be shared with either (or both) of Computer Science or Electrical Engineering Curricula. For instance,

a) there may be enough students of computer engineering to justify the provision of specialist courses devised solely for computer engineering students,

b) alternatively computer engineers may be required to attend classes offered from the Computer Science and Electronic Engineering curricula with additional selected classes being mounted specifically to address the specialist topics for computer engineering students; typically these will be classes addressing material that straddles the boundary between hardware and software and there ought to be at the very least one such class per year, ideally more,

c) additional possibilities also exist depending on local arrangements and circumstances.

Of these (a) above offers maximum flexibility. But regardless of these alternatives, the responsibility for the administration of courses in computer engineering have a duty to argue the case for high quality courses that meet the needs of their students.

7.2 Basic Computer Engineering Components

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

Introductory Courses and the Core

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

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

Intermediate Courses

By their very nature intermediate courses provide a bridge between introductory courses and advanced courses. They may well include core material but additionally material that falls outside the core. They will typically have as prerequisites introductory courses or indeed other intermediate courses.

Intermediate courses will typically be offered at second and third year level. Students may have a choice of computer engineering course but that choice is likely to be limited.

Advanced Courses

The term 'Advanced Course' is to be interpreted to mean courses whose content is substantially beyond the material of the core. The knowledge units give testimony to the rich set of possibilities that exist for these. Institutions will wish to orient such courses to their own areas of expertise, guided by the needs of students, the expertise of faculty members and the needs of the wider community. They will reflect leading edge developments and will be aligned with the stated orientation of the degree program. However, if specific core units are not included in the introductory and intermediate phase, the institution must then ensure that students acquire this material in advanced courses.

Institutions should give students a reasonable choice of advanced courses so that they can specialize in areas of choice.

7.2.1 Capstone Project
The culmination of the study of computer engineering should include a final year project in which it is required to demonstrate the use of a range of practices and techniques in solving a substantial problem.

7.2.2 Engineering Professional, Legal, and Ethical Issues

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

Fundamentally the teacher needs to be guided by a philosophy that is derived from an understanding and appreciation of the engineering ethos. In this way the whole approach to learning, teaching and assessment will be fashioned in a manner that reflects the fundamental philosophy and ethos that is required.

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

7.2.3 Assessment of student learning

There are a number of important considerations in the assessing of students’ learning above and beyond those that apply to all university learning.

- There is the issue of coursework; many topics lend themselves naturally to practical laboratory work. It is normally desirable to ensure that the practical work counts towards the final assessment; indeed some would take the view that a pass in the practical activity should be mandatory for a pass overall. All aspects of the practical activity must be of high quality

- Where there are sophisticated technical skills involved there should be recognition of properly time-tabled laboratories with support for the students to ensure that they are learning the material and acquiring effective skills.

Where transferable skills are being assessed there is merit in integrating this assessment with the assessment of computer science activity. Then the skills are seen in their natural setting and students are motivated to address them. An additional advantage of this approach is that it serves to reduce the assessment load; imaginative approaches to assessment are desirable.

7.3 Courses (Possibly) Outside Computer Engineering
Beyond the technical courses specifically on computer engineering, a number of other courses reflect material that needs to be present in the curriculum. For example, computer engineering students must learn a certain amount of mathematics. In this subsection we discuss various material that students must learn, but that will be, at least in many cases, taught outside of the department where computer engineering resides.

### 7.3.1 Mathematical 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 be introduced 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 structure 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. This dependency, moreover, should be reflected in the formal prerequisite structure.

Some material that is mathematical in nature, especially Discrete Structures (DS) lies in a boundary region between computer science and engineering on the one hand, and mathematics on the other hand, and may well be taught by computer engineering faculty. Other material, such as basic differential and integral calculus, will almost certainly be taught outside the department where computer engineering resides. However, in all cases, computer engineering programs must take responsibility for ensuring that students obtain the mathematics they need.

The Computer Science Volume discusses the special role of Discrete Structures (DS): material that is central to all students of computer science, and we hasten to add, computer engineering, and that can be considered as much as part of computer science and engineering as of mathematics. Accordingly, we both specify required DS knowledge in our BOK, and discuss it here, since we believe that this particular subject will be taught by computer science and/or engineering faculty at many institutions, but will also often be taught by the mathematics faculty. If an institution is fortunate enough to have a mathematics department that offers, for example, a discrete mathematics course appropriate for first-year students, that same course may well serve the needs of computer engineering. In many institutions, however, the computer engineering department must develop and teach these courses on their own. Yet another possibility is to integrate the teaching of DS—or other mathematics—with concepts from computer engineering, in particular to enhance student's interest.
The Computer Engineering Task Force makes the following recommendations with respect to the mathematical content of the computer engineering curriculum:

- **Discrete mathematics.** All students need exposure to the tools of discrete mathematics. All programs should include enough exposure to this area to cover the core topics in the DS area specified in the Computer Engineering BOK.

- **Differential and integral calculus.** The calculus is required to support such computer engineering material as communications theory, signals and systems, and analog electronics.

- Additional mathematics. Students should take additional mathematics to develop their sophistication in this area and to support classes in topics such as communications theory, security, signals and systems, analogue electronics. That mathematics might consist of courses in any number of areas including further calculus, differential equations, transform theory, linear algebra, numerical methods, geometry, number theory, or symbolic logic. The choice should depend on program objectives, institutional requirements, and the needs of the individual student.

- **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, all students should get at least some brief exposure to discrete and continuous probability, sampling distributions, estimation, hypothesis testing, and correlation and regression.

### 7.3.2 Scientific Method

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

To develop a firm understanding of the scientific method, students must have direct hands-on experience with hypothesis formulation, experimental design, hypothesis testing, and data analysis. While a curriculum may provide this experience in various ways (e.g., through a required physics sequence, or through a choice of courses in biology, chemistry or in physics) one 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 students must "do science", not just "read about science".

The Computer Engineering Task Force therefore makes the following recommendations about science:
• Students must develop an understanding of the scientific method and experience this mode of inquiry in courses that provide some exposure to laboratory work.

• Students may acquire their scientific perspective in any of a variety of domains, depending on program objectives and their area of interest.

### 7.3.3 Communication Skills

Students in computer engineering must be able to communicate ideas effectively in writing and in both formal and informal oral presentations.

Computer Engineering programs must, therefore, develop in their students the ability to present succinctly to a range of audiences (orally, electronically, and in writing) rational and reasoned arguments.

### 7.4 Degree Program Implementation: Strategies and examples

Institutions that adopt the CC2001 model will typically begin by choosing an implementation for the introductory phase and an implementation for the intermediate phase. From there, advanced courses that fit local conditions will be chosen.

#### 7.4.1 Course Considerations

As mentioned previously the precise courses area going to depend on the character of each individual program of study. But in broad terms the various courses will tend to be governed by considerations of the following kind.

**Introductory Courses**

At the initial stages it is appropriate to develop basic skills. Accordingly introductory courses will be required to address

- Basic skills in programming; various approaches to this have been outlined in the Computer Science volume and it is relevant to make reference to these alternative models
- Basic skills in the design and development of a range of electronic circuits, analogue systems and
- An understanding of the basic structure and organization of a variety of computer systems; including computer communications; this needs to address the basic electronics aspects, the chip aspects, and the software approach; this should serve to integrate the various aspects of the course and provide an overview of the...
discipline of computer engineering. Fundamentally the perspective of the computer system as a hierarchy of abstract machines is relevant

Intermediate Courses

Equipped with basic skills the intermediate courses seek to develop these further and to indicate how these are to be utilized in the design and the development of various components – either hardware, software, communications or hybrid systems. Again the choices here will depend heavily on the precise characteristics of the program of study.

In developing such courses it is important to be aware that skills such as programming required constant reinforcing. Thus it is typically not desirable to introduce students to programming and then drop programming for a complete semester.

Advanced Elective Courses

The Computer Engineering Task Force has identified a range of possible advanced elective courses which focus on material that, in keeping with the spirit of computer engineering, involves both hardware and software to an advanced level. Of course it is recognized that other courses may concentrate on specific aspects of hardware or software. In brief, computer engineering advanced courses might include the following:

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

7.4.2 Sample Curricula

In Appendix B we provide XXX sample implementations of complete computer engineering programs.
In order to provide a framework for the curriculum which seems to illustrate the ideas outlined above, the following will be assumed:

- there are two semesters each year, with a student studying a total of five modules per semester; each module is approximately 45 hours of contact time

- there are 3 computer engineering modules in the first year of study, 4 or 5 in the second year and 5 or 6 in each of the third and fourth years.

The above pattern is familiar to many US institutions, and indeed is common in many other parts of the world.
Chapter 8

Institutional Challenges

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

8.1 The need for local adaptation

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

- **The type of institution and the expectations for its degree programs.** Institutions vary enormously in the structure and scope of undergraduate degree requirements. A curriculum that works well at a small college in the United States may be completely inappropriate for a research university elsewhere in the world.
- **The range of postgraduate options that students pursue.** Institutions whose primary purpose is to prepare a skilled workforce for the computer engineering profession presumably have different curricular goals than those seeking to prepare research students for graduate study. Individual schools must ensure that the curriculum they offer gives students the necessary preparation for their eventual academic and career paths.
- **The preparation and background of entering students.** Students at different institutions—and often within a single institution—vary substantially in their level of preparation. As a result, computer engineering departments often need to tailor their introductory offerings so that they meet the needs of their students.
- **The faculty resources available to an institution.** The number of faculty in a computer engineering department may vary from as little as three or four at a small college to 100 or more at a large research university. The flexibility and options available in these smaller programs is obviously a great deal less. Therefore, faculty in smaller departments need to set priorities for how they will use their limited resources.
- **The interests and expertise of the faculty.** Individual curricula often vary according to the specific interests and knowledge base of the department, particularly at smaller institutions where expertise is concentrated in particular areas.
Creating a workable curriculum requires finding an appropriate balance among these factors, which will require different choices at every institution. There can be no single curriculum that works 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 that environment.

8.2 Principles for curriculum design

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

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

- 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 should indeed be excited by the prospect of doing so. One of the most important goals of a computer engineering program should be to produce students who are life-long learners.

- Curriculum design must be guided by the outcomes you hope to achieve. 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 the goals are met—provide the foundation for the entire curriculum. In the United States and elsewhere, 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 intend.

- The curriculum as a whole should maintain a consistent ethos that promotes innovation, creativity, and professionalism. Students respond best when they understand what it is expected 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.
• The curriculum must provide students with a culminating design experience that gives them a chance to apply their skills and knowledge to solve challenging problems. The culmination of an undergraduate computer engineering degree should include a project that requires students to use a range of practices and techniques in solving a substantial problem.

8.3 The need for adequate laboratory resources

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

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

8.4 Attracting and retaining faculty

One of the most daunting problems that computer engineering departments face is the problem of attracting qualified faculty. In computer engineering, there are often more advertised positions than 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.

8.5 Summary

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

[Abelson et al, 1985]

[ABET, 2000]


[ACM, 1965]

[ACM, 1968]

[ACM, 1978]

[ACM, 1999]

[ACM, 2001]

[APP, 2000]

[BCS, 1989a]

[BCS, 1989b]

[Beidler et al, 1985]

[Bennett, 1986]

[Bott et al, 1991]

[Carnegie, 1992]

[COSINE, 1967]

[CSAB, 1986]

[CSAB, 2000]

[CSTB, 1994]

[CSTB, 1999]

[Curtis, 1983]

[Davis et al, 1997]
Gordon B. Davis, John T. Gorgone, J. Daniel Couger, David L. Feinstein, and Herbert E. Longnecker, Jr. IS’97 model curriculum and guidelines for undergraduate degree programs in information systems. Association of Information...
[Denning et al, 1989]

[Denning, 1998]

[Denning, 1999]

[EAB, 1983]

[EAB, 1986]

[EC, 1977]

[Fellows et al, 2002]

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

[Fledermann, 2000]

[Gibbs et al, 1986]

[Giladi, 1999]
[Gorgone et al, 2000]

[Gorgone et al, 2002]

[IEEE, 2001]

[Kelemen et al, 1999]

[Koffman et al, 1984]

[Koffman et al, 1985]

[Lee and Messerschmitt, 1998]

[Lidtke et al, 1999]

[Martin et al, 1996]
[Mulder, 1975]

[Mulder and Dalphin, 1984]

[Mulder and van Weert, 1998]

[Myers and Walker, 1998]

[NACE, 2001]

[Neumann, 1995]

[Nordheden and Hoeflich, 1999]

[NSF, 1996]

[NTIA, 1999]

[Nunamaker et al, 1982]

[Oklobdzija, 2002]

55

[OTA, 1988]

[Pau1k et al, 1995]

[QAA, 2000]

[Ralston and Shaw, 1980]

[Richard et al, 1999]

[Roberts et al, 2001]

[Roberts et al, 1995]

[Roberts, 1999]

[SAC, 1967]

[SEEPP, 1998]
[Shaw, 1985]

[Shaw, 1991]

[Shaw, 1992]

[SIGCHI, 1992]

[SWEBOK, 2001]

[Tucker et al, 1991]

[UKQAA, 2000]

[Walker and Schneider, 1996]

[Zadeh, 1968]
6.2 Degree Program Design

There are a variety of different approaches to the difficult task of designing a curriculum. Apart from addressing the general requirements identified above, the topics identified within the core need to be included and advanced courses need to be on offer to enhance the curriculum. Additional constraints will appear in the form of local needs (institutional or regional), responding to the needs of an increasingly diverse student population and producing a curriculum with which faculty can identify.

Considerations of this kind lead to institutions offering courses which exhibit a particular character reflecting the anticipated characteristics of graduates. This results in a set of requirements which have been prescribed and they need to be realised through a number of classes which themselves will have a particular orientation. An important level of checking should ensure that by following the prescribed set of classes and experiences the more global course requirements are addressed.

In designing courses the following principles of course design will be applied:

- computer engineering is a discipline in its own right

- that discipline is characterised by a combination of knowledge, theory, practice and an appreciation of applications; practice is guided by theory and the whole is characterised by a professional approach

- computer engineering is a vibrant and fast changing discipline; graduates must be prepared to keep up-to-date with modern developments and should be excited by the prospect of doing so and indeed benefiting from these developments

- study of computer engineering is characterised by a problem solving approach with creativity featuring heavily and innovation being cherished

- within each degree program there are themes that should be developed, with new graduates being at the forefront of knowledge on a number of these

The culmination of the study of computer science should include a final year project in which it is required to demonstrate the use of a range of practices and techniques in solving a substantial problem.

A variety of possible degree programs are perfectly possible 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
Beyond concerns for mathematics and the scientific method the curriculum must address the elements of the engineering ethos as well as professional, legal and ethical issues with progression and integration taking place within these elements as well as within the technical domain. Addressing this vast array of requirements presents a complex task. If the various requirements are treated separately and in an undisciplined fashion the result will be less than satisfactory.

Fundamentally the teacher needs to be guided by a philosophy that is derived from an understanding and appreciation of the engineering ethos. In this way the whole approach to learning, teaching and assessment will be fashioned in a manner that reflects the fundamental philosophy and ethos that is required.

Earlier, mention was made of the importance of giving attention to creativity and innovation in a computer engineering context. It is worth remarking that certain approaches to the other important matter of professional, legal and ethical matters can have the highly undesirable effect of stifling beneficial innovation. Teachers need to recognise this and indeed take positive steps to counter such trends. It is most important to ensure that the balance is heavily in favour of beneficial innovation and creativity. Another aspect that is not addressed is the difficulty area of assessment. There are a number of important matters to mention in relation to this:

There is the issue of coursework; many topics lend themselves naturally to practical laboratory work. It is normally desirable to ensure that the practical work counts towards the final assessment; indeed some would take the view that a pass in the practical activity should be mandatory for a pass overall. All aspects of the practical activity must be of high quality.

Where there are sophisticated technical skills involved there should be recognition of properly time-tabled laboratories with support for the students to ensure that they are learning the material and acquiring effective skills.

Where transferable skills are being assessed there is merit in integrating this assessment with the assessment of computer science activity. Then the skills are seen in their natural setting and students are motivated to address them. An additional advantage of this approach is that it serves to reduce the assessment load; imaginative approaches to assessment are desirable.

Implementation of any course in computer engineering will be governed by many local factors. For instance,
a) there may be enough students of computer engineering to justify the provision of specialist courses devised solely for computer engineering students.

b) alternatively computer engineers may be required to attend classes offered from the Computer Science and Electronic Engineering curricula with additional selected classes being mounted specifically to address the specialist topics for computer engineering students; typically these will be classes addressing material that straddles the boundary between hardware and software and there ought to be at the very least one such class per year, ideally more.

c) additional possibilities also exist depending on local arrangements and circumstances. Of these (a) above offers maximum flexibility. But regardless of these alternatives, the responsibility for the administration of courses in computer engineering have a duty to argue the case for high quality courses that meet the needs of their students. Collectively the learning outcomes (taken over the entire set of modules in a degree program) must address the overall aims and objectives of the course and contribute to a view of the profile of graduates.

6.5.1 Strategies