Appendix A

The Computer Engineering Body of Knowledge

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

A.1 Introduction

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

CE-ALG*	Algorithms
CE-CAO	Computer Architecture and Organization
CE-CSE	Computer Systems Engineering
CE-CSG	Circuits and Signals
CE-DBS	Database Systems
CE-DIG	Digital Logic
CE-DSC*	Discrete Structures
CE-DSP	Digital Signal Processing
CE-ELE	Electronics
CE-ESY	Embedded Systems
CE-HCI*	Human-Computer Interaction
CE-NWK	Computer Networks
CE-OPS*	Operating Systems
CE-PRF*	Programming Fundamentals
CE-PRS	Probability and Statistics
CE-SPR*	Social and Professional Issues
CE-SWE*	Software Engineering
CE-VLS	VLSI Design and Fabrication

* Consult the CC2001 Computer Science Report for more detail

A.2 Structure of the Body of Knowledge

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

- The highest level of the hierarchy is the *knowledge area*, which represents a particular disciplinary sub-field. A three-letter abbreviated tag identifies each area, such as CE-DIG for “Digital Logic” and CE-CAO for “Computer Architecture and Organization.”

- Each knowledge area is broken down into smaller divisions called *knowledge units*, which represent individual thematic modules within an area. A numeric suffix added to the area name identifies each knowledge unit. For example, CE-CAO3 is a knowledge unit on "Memory System Organization and Architecture" within the CE-CAO knowledge area.
- A set of *topics*, which are the lowest level of the hierarchy, further subdivides each knowledge unit. A group of learning outcomes addresses the related technical skills associated with each knowledge unit. Section 4.3 expands the discussion on learning outcomes.

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

A.3 Core and Elective Units

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

The following points are emphasized:

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

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

A.4 Time Required to Cover a Knowledge Unit

For consistency with the Computer Science Report and earlier curriculum reports, we have chosen to express time in **hours**, corresponding to the in-class time required to present that material in a traditional lecture-oriented format. To dispel any potential confusion, however, it is important to underscore the following observations about the use of lecture hours as a measure.

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

presumably take roughly five times as much time to cover as a one-hour unit, independent of the teaching style.

The hours specified do not include time spent outside of a class. The time assigned to a unit does not include the instructor's reparation time of the time students spend outside of class. As a general guideline, the amount of out-of-class work for a student is approximately two to three times the in-class time. Thus, a unit that requires three hours of instruction typically entails a total of nine to twelve hours (three in class and six to nine outside).

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

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

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

A.5 Summary of the Computer Engineering Body of Knowledge

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

Table A-1 The Computer Engineering Body of Knowledge

Computer Engineering Knowledge Areas and Units	
CE-ALG Algorithms [30 core hours] CE-ALG0 History and overview [1] *CE-ALG1 Basic algorithmic analysis [4] *CE-ALG2 Algorithmic strategies [8] *CE-ALG3 Computing algorithms [12] *CE-ALG4 Distributed algorithms [3] *CE-ALG5 Algorithmic complexity [2] *CE-ALG6 Basic computability theory	**CE-CAO Computer Architecture and Organization** [63 core hours] CE-CAO0 History and overview [1] CE-CAO1 Fundamentals of computer architecture [10] CE-CAO2 Computer arithmetic [3] CE-CAO3 Memory system organization and architecture [8] CE-CAO4 Interfacing and communication [10] CE-CAO5 Device subsystems [5] CE-CAO6 Processor systems design [10] CE-CAO7 Organization of the CPU [10] CE-CAO8 Performance [3] CE-CAO9 Distributed system models [3] CE-CAO10 Performance enhancements
CE-CSE Computer Systems Engineering [18 core hours] CE-CSE0 History and overview [1] CE-CSE1 Life cycle [2] CE-CSE2 Requirements analysis and elicitation [2] CE-CSE3 Specification [2] CE-CSE4 Architectural design [3] CE-CSE5 Testing [2] CE-CSE6 Maintenance [2] CE-CSE7 Project management [2] CE-CSE8 Concurrent (hardware/software) design [2] CE-CSE9 Implementation CE-CSE10 Specialized systems CE-CSE11 Reliability and fault tolerance	**CE-CSG Circuits and Signals** [43 core hours] CE-CSG0 History and overview [1] CE-CSG1 Electrical Quantities [3] CE-CSG2 Resistive Circuits and Networks [9] CE-CSG3 Reactive Circuits and Networks [12] CE-CSG4 Frequency Response [9] CE-CSG5 Sinusoidal Analysis [6] CE-CSG6 Convolution [3] CE-CSG7 Fourier Analysis CE-CSG8 Filters CE-CSG9 Laplace Transforms
CE-DBS Database Systems [5 core hours] CE-DBS0 History and overview [1] *CE-DBS1 Database systems [2] *CE-DBS2 Data modeling [2] *CE-DBS3 Relational databases *CE-DBS4 Database query languages *CE-DBS5 Relational database design *CE-DBS6 Transaction processing *CE-DBS7 Distributed databases *CE-DBS8 Physical database design	**CE-DIG Digital Logic** [57 core hours] CE-DIG0 History and overview [1] CE-DIG1 Switching theory [6] CE-DIG2 Combinational logic circuits [4] CE-DIG3 Modular design of combinational circuits [6] CE-DIG4 Memory elements [3] CE-DIG5 Sequential logic circuits [10] CE-DIG6 Digital systems design [12] CE-DIG7 Modeling and simulation [5] CE-DIG8 Formal verification [5] CE-DIG9 Fault models and testing [5] CE-DIG10 Design for testability
CE-DSP Digital Signal Processing [17 core hours] CE-DSP0 History and overview [1] CE-DSP1 Theories and concepts [3] CE-DSP2 Digital spectra analysis [1] CE-DSP3 Discrete Fourier transform [7] CE-DSP4 Sampling [2] CE-DSP5 Transforms [2] CE-DSP6 Digital filters [1] CE-DSP7 Discrete time signals CE-DSP8 Window functions CE-DSP9 Convolution CE-DSP10 Audio processing CE-DSP11 Image processing	**CE-ELE Electronics** [40 core hours] CE-ELE0 History and overview [1] CE-ELE1 Electronic properties of materials [3] CE-ELE2 Diodes and diode circuits [5] CE-ELE3 MOS transistors and biasing [3] CE-ELE4 MOS logic families [7] CE-ELE5 Bipolar transistors and logic families [4] CE-ELE6 Design parameters and issues [4] CE-ELE7 Storage elements [3] CE-ELE8 Interfacing logic families and standard buses [3] CE-ELE9 Operational amplifiers [4] CE-ELE10 Circuit modeling and simulation [3] CE-ELE11 Data conversion circuits CE-ELE12 Electronic voltage and current sources CE-ELE13 Amplifier design CE-ELE14 Integrated circuit building blocks
CE-ESY Embedded Systems [20 core hours] CE-ESY0 History and overview [1] CE-ESY1 Embedded microcontrollers [6] CE-ESY2 Embedded programs [3] CE-ESY3 Real-time operating systems [3] CE-ESY4 Low-power computing [2] CE-ESY5 Reliable system design [2] CE-ESY6 Design methodologies [3] CE-ESY7 Tool support CE-ESY8 Embedded multiprocessors CE-ESY9 Networked embedded systems CE-ESY10 Interfacing and mixed-signal systems	**CE-HCI Human-Computer Interaction** [8 core hours] CE-HCI0 History and overview [1] *CE-HCI1 Foundations of human-computer interaction [2] *CE-HCI2 Graphical user interface [2] *CE-HCI3 I/O technologies [1] *CE-HCI4 Intelligent systems [2] *CE-HCI5 Human-centered software evaluation *CE-HCI6 Human-centered software development *CE-HCI7 Interactive graphical user-interface design *CE-HCI8 Graphical user-interface programming *CE-HCI9 Graphics and visualization *CE-HCI10 Multimedia systems

CE-NWK Computer Networks [21 core hours]
- CE-NWK0 History and overview [1]
- CE-NWK1 Communications network architecture [3]
- CE-NWK2 Communications network protocols [4]
- CE-NWK3 Local and wide area networks [4]
- CE-NWK4 Client-server computing [3]
- CE-NWK5 Data security and integrity [4]
- CE-NWK6 Wireless and mobile computing [2]
- CE-NWK7 Performance evaluation
- CE-NWK8 Data communications
- CE-NWK9 Network management
- CE-NWK10 Compression and decompression

CE-OPS Operating Systems [20 core hours]
- CE-OPS0 History and overview [1]
- *CE-OPS1 Design principles [5]
- *CE-OPS2 Concurrency [6]
- *CE-OPS3 Scheduling and dispatch [3]
- *CE-OPS4 Memory management [5]
- *CE-OPS5 Device management
- *CE-OPS6 Security and protection
- *CE-OPS7 File systems
- *CE-OPS8 System performance evaluation

CE-PRF Programming Fundamentals [39 core hours]
- CE-PRF0 History and overview [1]
- *CE-PRF1 Programming Paradigms [5]
- *CE-PRF2 Programming constructs [7]
- *CE-PRF3 Algorithms and problem-solving [8]
- *CE-PRF4 Data structures [13]
- *CE-PRF5 Recursion [5]
- *CE-PRF6 Object-oriented programming
- *CE-PRF7 Event-driven and concurrent programming
- *CE-PRF8 Using APIs

CE-SPR Social and Professional Issues [16 core hours]
- CE-SPR0 History and overview [1]
- *CE-SPR1 Public policy [2]
- *CE-SPR2 Methods and tools of analysis [2]
- *CE-SPR3 Professional and ethical responsibilities [2]
- *CE-SPR4 Risks and liabilities [2]
- *CE-SPR5 Intellectual property [2]
- *CE-SPR6 Privacy and civil liberties [2]
- *CE-SPR7 Computer crime [1]
- *CE-SPR8 Economic issues in computing [2]
- *CE-SPR9 Philosophical frameworks

CE-SWE Software Engineering [13 core hours]
- CE-SWE0 History and overview [1]
- *CE-SWE1 Software processes [2]
- *CE-SWE2 Software requirements and specifications [2]
- *CE-SWE3 Software design [2]
- *CE-SWE4 Software testing and validation [2]
- *CE-SWE5 Software evolution [2]
- *CE-SWE6 Software tools and environments [2]
- *CE-SWE7 Language translation
- *CE-SWE8 Software project management
- *CE-SWE9 Software fault tolerance

CE-VLS VLSI Design and Fabrication [10 core hours]
- CE-VLS0 History and overview [1]
- CE-VLS1 Electronic properties of materials [2]
- CE-VLS2 Function of the basic inverter structure [3]
- CE-VLS3 Combinational logic structures [1]
- CE-VLS4 Sequential logic structures [1]
- CE-VLS5 Semiconductor memories and array structures [2]
- CE-VLS6 Chip input/output circuits
- CE-VLS7 Processing and layout
- CE-VLS8 Circuit characterization and performance
- CE-VLS9 Alternative circuit structures/low power design
- CE-VLS10 Semi-custom design technologies
- CE-VLS11 ASIC design methodology

Mathematics Knowledge Areas and Units

CE-DSC Discrete Structures [33 core hours]
- CE-DSC0 History and overview [1]
- *CE-DSC1 Functions, relations, and sets [6]
- *CE-DSC2 Basic logic [10]
- *CE-DSC3 Proof techniques [6]
- *CE-DSC4 Basics of counting [4]
- *CE-DSC5 Graphs and trees [4]
- *CE-DSC6 Recursion [2]

CE-PRS Probability and Statistics [33 core hours]
- CE-PRS0 History and overview [1]
- CE-PRS1 Discrete probability [6]
- CE-PRS2 Continuous probability [6]
- CE-PRS3 Expectation [4]
- CE-PRS4 Stochastic Processes [6]
- CE-PRS5 Sampling distributions [4]
- CE-PRS6 Estimation [4]
- CE-PRS7 Hypothesis tests [2]
- CE-PRS8 Correlation and regression

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

A.6 Comments on Knowledge Areas

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

A.6.1 Comments on Algorithms

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

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

A.6.2 Comments on Computer Architecture and Organization

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

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

A.6.3 Comments on Computer Systems Engineering

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

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

A.6.4 Comments on Circuits and Signals

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

A.6.5 Comments on Database Systems

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

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

A.6.6 Comments on Digital Logic

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

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

A.6.7 Comments on Discrete Structures

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

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

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

A.6.8 Comments on Digital Signal Processing

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

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

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

A.6.9 Comments on Electronics

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

A.6.10 Comments on Embedded Systems

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

A.6.11 Comments on Human-Computer Interaction

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

Emphasis is placed on understanding human reactions to displays of various kinds and on human behavior in the context of interactive objects. Based on these, students need to understand the principles associated with the evaluation of interfaces including those that embody interaction. Students need to know the principles and guidelines that reflect best practice in the design, development, and maintenance of interfaces for multiple types of systems.

A.6.12 Comments on Computer Networks

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

A.6.13 Comments on Operating Systems

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

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

A.6.14 Comments on Programming Fundamentals

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

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

A.6.15 Comments on Probability and Statistics

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

A.6.16 Comments on Social and Professional Issues