Chapter 2

Computer Engineering as a Discipline

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

2.1 Background

Computer engineering is a discipline that embodies the science and technology of design, construction, implementation, and maintenance of software and hardware components of modern computing systems, computer-controlled equipment, and networks of intelligent devices. Traditionally, computer engineering is some combination of both electrical engineering (EE) and computer science (CS). It has evolved over the past four decades as a separate discipline, although intimately related to computer science and electrical engineering. Computer engineering is solidly grounded in the theories and principles of computing, mathematics, science, and engineering and it applies these theories and principles to solve technical problems through the design of computing hardware, software, networks, and processes.

Historically, the field of computer engineering has been widely viewed as “designing computers.” In fact, the design of computers themselves has been the province of relatively few highly skilled engineers whose goal was to push forward the limits of computer and microelectronics technology. The successful miniaturization of silicon devices and their increased reliability as system building blocks and complete systems on chips have created an environment in which computers have become pervasive and replaced more conventional electronic devices. These applications manifest themselves in the proliferation of mobile smart phones, tablet computers, multimedia and location-aware devices, wireless networks, and similar products. Computer engineering also reveals itself in the myriad of applications involving embedded systems, namely those computing systems that appear in applications such as automobiles, control systems, major appliances, and the internet of things.

Increasingly, computer engineers are involved in the design of computer-based systems to address highly specialized and specific application needs. Computer engineers work in most industries, including the computer, automobile, aerospace, telecommunications, power production, manufacturing, defense, and electronics industries. They design high-tech devices ranging from tiny microelectronic integrated-circuit chips, to powerful systems that utilize those chips and efficient telecommunication systems that interconnect those systems. Computer engineers also work on distributed computing environments—local and wide area networks, wireless networks, internets, intranets—and embedded computer systems—such as in aircraft, spacecraft, and automobile control systems where they perform various functions. A wide array of complex technological systems, such as power generation and distribution systems and modern processing and manufacturing plants, rely on computer systems developed and designed by computer engineers.

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

2.3.1 Distinctions

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

- the ability to design computers, computer-based systems, and networks that include both hardware and software as well as their integration to solve novel engineering problems, subject to trade-offs involving a set of competing goals and constraints—in this context, "design" refers to a level of ability beyond "assembling" or "configuring" systems
- a breadth of knowledge in mathematics and engineering sciences, associated with the broader scope of engineering and beyond that narrowly required for the field
- acquisition and maintenance of a preparation for professional practice in engineering

Other related disciplines can be described as follows.

- Electrical engineering spans a wide range of areas, including bioengineering, power engineering, electronics, telecommunications, and digital systems. Related to the field of computer engineering, electrical engineers concern themselves primarily with the physical aspects of electronics including circuits, signal analysis, and microelectronic devices.
- Computer scientists concern themselves primarily with the theoretical and algorithmic aspects of computing with a focus on the theoretical underpinnings of computing.
- Software engineers have a focus on the principles underlying the development and maintenance of correct (often large-scale) software throughout its lifecycle. Information systems specialists encompass the acquisition, deployment, and management of information resources for use in organizational processes.

- Information technology specialists focus on meeting the needs of users within an organizational and societal context through the selection, creation, application, integration, and administration of computing technologies.
- Computer engineering technologists support engineers by installing and operating computer-based products, and maintaining those products.

Table 4.1: Types of computer engineering laboratories

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

Table 4.2 shows additional laboratory options that programs could consider.

Table 4.2: Suggested Additional Computer Engineering Laboratories

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

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

4.4.2 Software considerations

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

Table 4.3: Suggestions for possible software applications

Design modeling and simulation
- Circuit-level (e.g., SPICE)
- Gate-level (schematic entry)
- Digital systems (e.g., VHDL, Verilog)
- Analog/mixed-signal circuits (e.g., VHDL-AMS, Verilog-AMS)
- System-level design (e.g., System Verilog, System C)

Digital hardware prototyping
- FPGA/CPLD development suite
- Design file entry and management
- Component/IP library support
- Device programming
- Interactive debugging

Microcontroller system design
- Integrated development environment (IDE)
  - Design entry/management
  - Library support
- Compilers, assemblers, linkers
- Processor simulators/emulators
- Device programming
- In-circuit test/debug

Mathematics packages
- Problem solving
- Data analysis
- Modeling and simulation

Software development
- Integrated development environment (IDE)
  - Design entry and management
  - Library support
- Compilers (e.g., C, C++, C#, Java, Python)
- Operating system support
- Source-level debugging
- Platform support (e.g., smartphone, tablet)

Integrated circuit/ASIC design
- Design capture and simulation
- Synthesis
- Physical layout
  - graphical layout editor
  - automated layout
- Design verification (design rules, layout vs. schematic, parameter extraction)
- Design for testability, automatic test pattern generation

Printed circuit board (PCB) design

Computer-aided design and modeling (CAD tools)

Laboratory automation and instrumentation (e.g., IntuiLink or LabVIEW software)

General computing/productivity
- Web browser
- Email
- Office suite
- PDF reader/editor
- Illustration/photo viewer/editor
- Multimedia players/editors
- File compression/ decompression
- File transfer (FTP, SCP, SFTP)
- Terminal emulator, remote login, secure shell, X Window client

System engineering tools
- Project management (e.g., GANTT or PERT charts)
- Requirements and specifications management (e.g., UML tools)

Other tools
- Robotics software development
- Semiconductor device and process modeling (e.g., TCAD)
- Microwave, RF and other high-speed/high-frequency design
- Electromagnetic field simulation
- MEMS design

4.4.3 Open-ended laboratories

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

The CE2004 report contains the two mathematics knowledge areas of discrete structures and probability and statistics. To this, the CE2016 BoK adds two additional mathematics knowledge areas—analysis of continuous functions and linear algebra. Most computer engineering programs already include coverage of these knowledge areas, and the inclusion now is simply an acknowledgement of that fact. The reasoning for including these four mathematics knowledge areas is offered here.

- *Discrete structures*: All students need knowledge of the mathematical principles of discrete structures and their related tools. All programs should include enough exposure to this area to cover the core learning outcomes specified in the computer engineering body of knowledge.
- *Analysis of continuous functions*: The calculus and differential equations are required to support such computer engineering material as communications theory, signals and systems, and analog electronics—the analysis of continuous functions is fundamental to all engineering programs.
- *Probability and statistics*: This mathematical area underpins considerations of reliability, safety, performance, dependence, and various other concepts of concern to the computer engineer. Many programs will have students take an existing course in probability and statistics; some programs may allow some students to study less than a full semester course in the subject. Regardless of the implementation, all students should get at least some brief exposure to discrete and continuous probability, stochastic processes, sampling distributions, estimation, hypothesis testing, and correlation and regression, as specified in the computer engineering body of knowledge.
- *Linear algebra*: Linear algebra is required for solving networks of equations describing voltage/current relationships in basic circuits, and is used in computer engineering application areas such as computer graphics and robotics.

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

6.4.2 Science Requirements

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

Computer engineering students need knowledge of natural sciences, such as physics and chemistry. Basic physics concepts in electricity and magnetism form the basis for much of the underlying electrical engineering content in the body of knowledge. Other science courses, such as biology, are relevant to specific application areas in which computer engineers may specialize. The precise nature of the natural science requirement will vary, based on institutional and programs needs and resources.

To develop a firm understanding of the scientific method, students must have direct hands-on experience with hypothesis formulation, experimental design, hypothesis testing, and data analysis. While a curriculum may provide this experience as part of the natural science coursework, another way of addressing this is through appropriate courses in computer engineering itself. For example, considerations of the user interface provide a rich vein of experimental situations.

It is vital that computer engineering students “do science” and not just “read about science” in their education. The overall objectives of this element of the curriculum include the following.

- Students should acquire knowledge of the natural sciences underlying computer engineering and relevant application areas.
- Students must develop an understanding of the scientific method and gain experience in this mode of inquiry through courses that provide some exposure to laboratory work, including data collection and analysis.
- Students may acquire their scientific perspective in any of a variety of domains, depending on program objectives and their area of interest.

6.4.3 Other Requirements

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

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

Computer Engineering Knowledge Areas and Units

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

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

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

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

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

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

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

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

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

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

Table 3.3: Related CE Mathematics
(120 Core Hours)

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

3.3.3 The role of software

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

A.5 Knowledge Areas and Knowledge Units

CE-CAE Circuits and Electronics

[50 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-CAE Core Knowledge Units

CE-CAE-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

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

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-CAE-3 Electrical quantities and basic elements

Minimum core coverage time: 4 hours

Core Learning Outcomes:

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

Elective Learning Outcomes:

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

CE-CAE-4 Electrical circuits

Minimum core coverage time: 11 hours

Core Learning Outcomes:

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

Elective Learning Outcomes:

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

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

Minimum core coverage time: 7 hours

Core Learning Outcomes:

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

Elective Learning Outcomes:

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

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

Minimum core coverage time: 12 hours

Core Learning Outcomes:

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

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

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

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

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

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

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

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

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

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

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

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

Elective Learning Outcomes:

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

CE-CAE Supplementary Knowledge Units

CE-CAE-11 Design parameters and issues
Supplementary

Elective Learning Outcomes:

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

CE-CAE-12 Circuit modeling and simulation methods
Supplementary

Elective Learning Outcomes:

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

CE-CAL Computing Algorithms

[30 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-CAL Core Knowledge Units

CE-CAL-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

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

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

CE-CAL-3 Basic algorithmic analysis

Minimum core coverage time: 4 hours

Core Learning Outcomes:

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

CE-CAL-4 Algorithmic strategies

Minimum core coverage time: 6 hours

Core Learning Outcomes:

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

CE-CAL-5 Classic algorithms for common tasks

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

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

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

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

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

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

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

CE-CAL Supplementary Knowledge Units

CE-CAL-9 Scheduling algorithms
Supplementary

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

CE-CAL-10 Basic computability theory
Supplementary

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

CE-CAO Computer Architecture and Organization

[60 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-CAO Core Knowledge Units

CE-CAO-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

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

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

Elective Learning Outcome:

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

CE-CAO-3 Instruction set architecture

Minimum core coverage time: 10 hours

Core Learning Outcomes:

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

Elective Learning Outcome:

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

CE-CAO-4 Measuring performance

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-CAO-5 Computer arithmetic

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

Elective Learning Outcomes:

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

CE-CAO-6 Processor organization

Minimum core coverage time: 10 hours

Core Learning Outcomes:

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

Elective Learning Outcomes:

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

CE-CAO-7 Memory system organization and architecture

Minimum core coverage time: 9 hours

Core Learning Outcomes:

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

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

Elective Learning Outcome:

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

CE-CAO-8 Input/output interfacing and communication

Minimum core coverage time: 7 hours

Core Learning Outcomes:

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

CE-CAO-9 Peripheral subsystems

Minimum core coverage time: 7 hours

Core Learning Outcomes:

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

CE-CAO-10 Multi/Many-core architectures

Minimum core coverage time: 5 hours

Core Learning Outcomes:

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

CE-CAO-11 Distributed system architectures

Minimum core coverage time: 4 hours

Core Learning Outcomes:

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

Elective Learning Outcomes:

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

CE-DIG Digital Design

[50 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-DIG Core Knowledge Units

CE-DIG-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

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

Minimum core coverage time: 2 hours

Core Learning Outcomes:

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

CE-DIG-3 Number systems and data encoding

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-DIG-4 Boolean algebra applications

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-DIG-5 Basic logic circuits

Minimum core coverage time: 6 hours

Core Learning Outcomes:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

CE-DIG Supplementary Knowledge Units

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

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

CE-ESY Embedded Systems

[40 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-ESY Core Knowledge Units

CE-ESY-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

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

Minimum core coverage time: 2 hours

Core Learning Outcomes:

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

CE-ESY-3 Characteristics of embedded systems

Minimum core coverage time: 2 hours

Core Learning Outcomes:

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

CE-ESY-4 Basic software techniques for embedded applications

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

CE-ESY Supplementary Knowledge Units

CE-ESY-13 Computing platforms for Embedded Systems
Supplementary

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

CE-NWK Computer Networks

[20 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-NWK Core Knowledge Units

CE-NWK-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

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

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

CE-NWK-3 Network architecture

Minimum core coverage time: 4 hours

Core Learning Outcomes:

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

CE-NWK-4 Local and wide area networks

Minimum core coverage time: 4 hours

Core Learning Outcomes:

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

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

Elective Learning Outcomes:

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

CE-NWK-5 Wireless and mobile networks

Minimum core coverage time: 2 hours

Core Learning Outcomes:

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

Elective Learning Outcome:

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

CE-NWK-6 Network protocols

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-NWK-7 Network applications

Minimum core coverage time: 2 hours

Core Learning Outcomes:

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

Elective Learning Outcomes:

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

CE-NWK-8 Network management

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-NWK Supplementary Knowledge Units

CE-NWK-9 Data communications

Supplementary

Elective Learning Outcomes:

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

CE-NWK-10 Performance evaluation

Supplementary

Elective Learning Outcomes:

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

CE-NWK-11 Wireless sensor network

Supplementary

Elective Learning Outcomes:

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

CE-PPP Preparation for Professional Practice

[20 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-PPP Core Knowledge Areas

CE-PPP-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

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

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

CE-PPP-3 Effective communication strategies

Minimum core coverage time: 2 hours

Core Learning Outcomes:

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

CE-PPP-4 Interdisciplinary team approaches

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

CE-PPP-5 Philosophical frameworks and cultural issues

Minimum core coverage time: 2 hours

Core Learning Outcomes:

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

CE-PPP-6 Engineering solutions and societal effects

Minimum core coverage time: 2 hours

Core Learning Outcomes:

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

CE-PPP-7 Professional and ethical responsibilities

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-PPP-8 Intellectual property and legal issues

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

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

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

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

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

CE-PPP Supplementary Knowledge Units

CE-PPP-11 Tradeoffs in professional practice
Supplementary

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

CE-SEC Information Security

[20 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-SEC Core Knowledge Units

CE-SEC-1 History and overview

Minimum core coverage time: 2 hours

Core Learning Outcomes:

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

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

Minimum core coverage time: 2 hours

Core Learning Outcomes:

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

CE-SEC-3 Data security and integrity

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

CE-SEC-4 Vulnerabilities: technical and human factors

Minimum core coverage time: 4 hours

Core Learning Outcomes:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

CE-SGP Signal Processing

[30 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-SGP Core Knowledge Units

CE-SGP-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

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

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-SGP-3 Convolution

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-SGP-4 Transform analysis

Minimum core coverage time: 5 hours

Core Learning Outcomes:

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

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

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

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

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

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

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

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

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

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

CE-SGP Supplementary Knowledge Units

CE-SGP-9 Window functions
Supplementary

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

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

CE-SGP-10 Multimedia processing
Supplementary

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

CE-SGP-11 Control system theory and applications
Supplementary

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

CE-SPE Systems and Project Engineering

[35 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-SPE Core Knowledge Units

CE-SPE-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

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

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-SPE-3 Project management

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

CE-SRM System Resource Management

[20 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-SRM Core Knowledge Units

CE-SRM-1 History and overview of operating systems

Minimum core coverage time: 1 hour

Core Learning Outcomes

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

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

Minimum core coverage time: 1 hour

Core Learning Outcomes

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

CE-SRM-3 Managing system resources

Minimum core coverage time: 8 hours

Core Learning Outcomes

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

CE-SRM-4 Real-time operating system design

Minimum core coverage time: 4 hours

Core Learning Outcomes

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

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

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

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

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

CE-SRM Supplementary Knowledge Units

CE-SRM-7 System performance evaluation
Supplementary

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

CE-SRM-8 Support for virtualization
Supplementary

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

CE-SWD Software Design

[45 core hours]

Area Scope

The knowledge units in this area collectively encompass the following:

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

CE-SWD Core Knowledge Units

CE-SWD-1 History and overview

Minimum core coverage time: 1 hour

Core Learning Outcomes:

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

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

Minimum core coverage time: 3 hours

Core Learning Outcomes:

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

CE-SWD-3 Programming constructs and paradigms

Minimum core coverage time: 12 hours

Core Learning Outcomes:

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

CE-SWD-4 Problem-solving strategies

Minimum core coverage time: 5 hours

Core Learning Outcomes:

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

Elective Learning Outcomes:

- Use dynamic programming to solve an appropriate problem.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

CE-SWD Supplementary Knowledge Units

CE-SWD-12 Using application programming interfaces
Supplementary

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

CE-SWD-13 Data mining
Supplementary

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

CE-SWD-14 Data visualization
Supplementary

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

Appendix C

Computer Engineering Laboratories

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

C.1 Circuits and Electronics

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

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

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

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

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

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

C.2 Computer Architecture Design

Typical Description: Techniques of design, simulation, and evaluation of a simple datapath and control using a hardware description language (e.g., VHDL or Verilog); assembly language programming on an emulated 32- or 64-bit microprocessor; implementation of an RTL model of an instruction set architecture in an FPGA.

Typical Configuration:

- computer with VHDL and/or Verilog modeling and simulation tools
- FPGA development board
- FPGA development suite to support the selected FPGA development board
- oscilloscope and logic analyzer to examine FPGA board outputs

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

C.3 Digital Logic Design

Typical Description: Experiments involving digital circuits of increasing complexity; combinational small-scale integration (SSI) and medium-scale integration (MSI) circuits; arithmetic and sequential circuits; analysis and synthesis of state machines.

Typical Configuration:

- breadboard for constructing digital circuits from SSI/MSI components
- oscilloscope and logic analyzer (separate instruments or integrated into the breadboard)
- FPGA development board
- computer with VHDL/Verilog modeling and simulation tools and FPGA development suite

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

C.4 Digital Signal Processing

Typical Description: Engage in hardware and software experiments showing digital signal processing principles and techniques; programming on DSP chips; real-time signal processing algorithms.

Typical Configuration:

- digital signal processor (DSP) development board/kit
- computer with DSP software development tools
- mixed-signal oscilloscope
- logic analyzer
- vector signal generator
- spectrum analyzer

Vendors for this equipment in 2016 include Texas Instruments, ARM, Tektronix, Agilent, Rhode & Schwarz, and others.

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

C.5 Digital Logic and System Design

Typical Description: Hierarchical, modular design of digital systems of increasing complexity; design, analysis, and synthesis of state machines; computer-aided digital system modeling, simulation, analysis, and synthesis; design implementation with programmable logic devices and/or FPGAs.

Typical Configuration:

- computer to host design tools
- VHDL and/or Verilog modeling and simulation tools
- FPGA development board
- proto board for interfacing peripheral components with the FPGA
- FPGA development suite to support the selected FPGA development board
- embedded processor soft core, or FPGA with an embedded processor hard core
- oscilloscope
- logic analyzer
- test pattern generator to provide FPGA inputs

Vendors for this equipment in 2016 include Digilent, Xilinx, Aldec, Altera, Tektronix, and others.

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

C.6 Embedded Systems

Typical Description: Experiments involving interfacing memory and peripheral devices to a microcomputer; design of software to control peripheral devices; integration of computer hardware and software for system control.

Typical Configuration:

- microcontroller development board/kit
- computer for hosting software tools
- integrated development environment for the selected microcontroller
- libraries of software modules to support selected peripheral devices
- powered breadboard for interfacing peripheral devices to the microcontroller
- oscilloscope
- logic analyzer
- multimeter
- triple output DC power supply

Vendors for this equipment in 2016 include ST Microelectronics, Digilent, Keil, Agilent, and others.

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

C.7 Engineering Introduction

Typical Description: Basic engineering skills and practice; students learn the basics of circuits, DC motors, and wireless communication; a design project involving some aspect of computer engineering (e.g., a radio-controlled car) culminates the experience; focus is on engineering design, teamwork, communication skills, and other related activities. This is often taught as a multi-disciplinary course.

Typical Configuration: Varies, based on the design project and orientation of the course.

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

C.8 Networking

Typical Description: Design and implementation of information networks based on requirements and devices such as routers and switches; applications of information networks for data, audio, and video communications; transmission media, modulation, error control, flow control, LANs, and Ethernet protocols; experiments on data communication signaling and error control; data transfer and software aspects of networks common in computing; implementation of servers and clients using various protocols.

Typical Configuration:

- multiple personal computers, to be integrated into a network
- network isolated from the institutional network
- configurable routers and/or switches
- network analyzer and/or software tools to measure network traffic and conditions

Vendors for this equipment in 2016 include Emona, Cisco, and others.

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

C.9 Software Design

Typical Description: Experience in software construction; testing, debugging, and associated tools; configuration management; low-level file and device I/O; systems and event-driven programming; languages to include C, C++, C#, Python, Ruby, Java and/or other appropriate languages in support of the computer engineering program.

Typical Configuration:

- modern computing platforms running widely-used modern operating systems
- IDE to manage project files and libraries
- compiler(s) and linker for language(s) used in the course
- source-level debugger
- documentation, presentation, file transfer, and other support tools
- mathematics package for analysis, simulation, and modeling
- database software

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