A Curriculum for Embedded System Engineering

RUDOLPH E. SEVIORA
University of Waterloo

The paper presents a curriculum for a 4-year undergraduate program in Embedded System Engineering (ESE). The curriculum was developed using a two-step approach. First, a body of education knowledge for Embedded System Engineering was defined. The body consists of sixteen knowledge areas. Each area is composed of several knowledge units, some designated as core and others as electives. The minimum lecture time for the core of each knowledge area is identified. The Body of Knowledge for Computer Engineering, developed by the IEEE-CS/ACM task force for Computing Curricula, was used as a reference. The education knowledge for ESE then served as the base for the development of the program curriculum. The curriculum has a strong mathematics and basic science base, an in-depth exposure to engineering science and design of systems implemented with digital hardware and software, and coverage of two prominent application areas of embedded systems. The curriculum core takes approximately 3 years of the program; the remaining part is elective.

Categories and Subject Descriptors: K.3.2 [**Computers and Education**]: Computer and Information Sciences Education Curriculum; C.3 [**Computer System Organization**]: Special-Purpose and Application-Based Systems—*Real-time and embedded systems*; J.7 [**Computer Applications**]: Computers and Other Systems—*Real time process control*

General Terms: Design

Additional Key Words and Phrases: Embedded system engineering, Embedded system engineering curriculum, Undergraduate engineering curriculum

1. INTRODUCTION

The design of an undergraduate curriculum for a discipline is a challenging undertaking. The designers can adopt a range of attitudes. At one end is the perspective of an industry manager who, when asked to rank the country's engineering programs, responded by saying that he had thought about this long and hard and concluded that there is little harm any university can do to a good mind. At the other end are the curricula that attempt to compress a large amount of material into the limited time available with the outcome aptly

Author's address: Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, N2L 3G1, Canada; email: seviora@swen.uwaterloo.ca.

characterized by [Bigelow 1865] "[the student] has spent much time and some labor in besieging the many doors of the temple of knowledge, without effecting an entrance to any of them."

An analogy with a sea illustrates the challenges. Underlying the discipline is a large body of knowledge (the sea of knowledge). The program curriculum attempts to enable graduates to navigate over the sea by building islands of knowledge to which the students are exposed. The available time limits the total area of islands. The hope is that a judicious positioning and sizing of islands would enable the graduates, possibly with reasonable additional effort, to get to most points in the sea. Unfortunately, the sea is constantly expanding and changing its shape, the currents are shifting, the islands are eroding, isolated islands are vanishing, and the importance of destinations is changing.

The field of computers and computing has been especially affected by the expansion of knowledge. The traditional academic response to dealing with the expansion is through increased specialization. The history of curricular efforts by ACM/IEEE-CS [1990] is instructive. Within 1 decade after the development of the single Computing Curricula 1991, a successor effort was initiated. Its outcome are five separate curricula addressing Computer Science, Computer Engineering, Software Engineering, Information Systems and Information Technology programs [Joint Task Force for Computing Curricula 2004]. At larger universities, this specialization may go even further. As an example, one university catalog lists seven programs with significant computing content, Computer Science, Computer Engineering, Software Engineering, Computational Science, Computational Mathematics, Mechatronics Engineering, and System Design Engineering [University of Waterloo Undergraduate Calendar 2004–5]. Since the underlying technology drivers have at least another decade of growth, there is expectation that the specialization process will continue and even accelerate as secondary effects take place.

Among the areas that have seen a staggering growth, both in terms of the underlying body of knowledge, as well as in the range of applications, is that of embedded systems. Embedded systems appear in a continuously expanding range of application areas, including transport, process control, electrical and electronic appliances, and telecommunications. This paper uses the term embedded system to designate a computer or a network of computers, possibly with attached digital hardware, which is a part of a larger system. The main distinguishing characteristics of embedded systems include tight coupling with their embedding, strict nonfunctional requirements, reactivity of operation, and, frequently, also long life.

Considering the extent and the growth of its knowledge base, it is increasingly noted that the engineering of embedded systems is becoming a discipline in its own right, with its own distinct set of fundamentals, engineering science and design, and engineering practice. This paper describes a curriculum aimed at educating embedded system engineers, i.e., professionals specializing in the design of embedded computers, their hardware, software and attached digital systems, and networks of such units.

The paper follows a two-step approach similar to that adopted by the ACM/IEEE-CS joint task force for Computing Curricula [Joint IEEE-CS/ACM

Task Force on Computing Curricula 2001, 2004, 2005]. Like the task force, it concentrates on the curricular segment that defines the professional profile of program graduates, namely engineering science and engineering design. The paper first delineates the body of education knowledge for Embedded System Engineering (ESE-BEK), i.e., the body of knowledge appropriate for undergraduate level. The body is structured hierarchically. It is divided into sixteen knowledge areas. Each area contains a number of units, smaller divisions that represent individual thematic modules in the area. Some units are core and some elective. The body of knowledge for Computer Engineering, described in the Computing Curricula for Computer Engineering (CCCE, [Joint IEEE-CS/ACM Task Force on Computing Curricula 2005]; [Nelson 2004]), is used as the starting point. The paper then assigns the minimum curricular coverage to the core of each knowledge area, using the number of lecture hours as the metric. Last, the paper presents a sample curriculum that meets the minimum coverage and accreditation requirements.

The author's views on the education knowledge base and the model curriculum were influenced by internal discussions at the author's institution about possible evolution of its computer engineering program. The author is greatly indebted to his colleagues and other discussion partners who helped to shape his perspectives. However, the views presented in this paper carry no institutional endorsement.

The paper is organized as follows. Section 2 discusses several curriculum design issues. Section 3 summarizes the IEEE-CS/ACM Computer Engineering curriculum, lists its knowledge areas and the curricular budget for their cores. Section 4 presents the body of education knowledge for Embedded System Engineering. Section 5 describes a sample curriculum for Embedded System Engineering, which is based on this body of knowledge and Section 6 offers concluding observations.

2. CURRICULUM DESIGN ISSUES

2.1 Accreditation and Industry Input

Accreditation. A basic requirement on an undergraduate program in engineering is that it be accreditable. Accreditation bodies define the basic structure of program curricula. Generally, they divide the curricular content into four major segments, Mathematics, Basic (Natural) Sciences, Engineering Science and Engineering Design, and Complementary Studies (humanities, social sciences, arts, management, engineering economics, and communication).

The stated curricular minima for these segments vary somewhat with the accreditation body. The minima of the U.S. Accreditation Board for Engineering and Technology (ABET), [2004–2005] are 1 year of Math and Science, 1.5 years of Engineering Science and Design, with 1.5 years for the rest in the case of 4-year programs. The minima of the Canadian Engineering Accreditation Board (CEAB, [2003]) are 1 year of Mathematics and Science, 2 years of Engineering Science and Design, and 0.5 year of Complementary Studies.

Accreditation criteria also typically require that the curriculum culminate in a significant design experience, which is based on the knowledge and skills

acquired in earlier course work and which preferably gives students exposure to the concepts of team work and project management.

In recent years, accreditation bodies have begun to put more emphasis on the outcomes of the educational experience. For ABET [2004–2005], the outcomes are categorized and could be grouped into the engineering design-, team work-, and society-related outcomes. An institution must have an assessment process to measure the outcomes and document its results.

It is worth noting that neither ABET nor CEAB specify the content of the Engineering Science and Design segment. Instead, they note that the content must be appropriate to the discipline. Also, while engineering is often described as applied science, the criteria do not stipulate that the curriculum include one or more major application areas for the discipline.

Industry Input. The curriculum design must also take into account the views of those who employ the program graduates. A good illustration of views related to embedded systems are the Curriculum Development Guidelines prepared by the Career Space, a consortium of eleven major information and communications technology companies operating in Europe [Career Space Consortium 2001]. The guidelines divide the curriculum into four segments and recommend the following distribution:

1. A scientific base of ∼30%;
2. A technology base of ∼30%;
3. An application base and systems thinking of ∼25%;
4. A personal and business skills element of up to ∼15%.

The guidelines also suggest that 3 months be allocated to each of team project work and practical experience through industry placement.

2.2 Technology Trends

The professional career of undergraduate students will last for 40 to 50 years. The curricular content considerations cannot be decoupled from the trends in technology and industry. While it is impossible to make predictions over such an interval of time, nearer term trends should be considered. An engineering curriculum must be forward looking, not an exercise in engineering nostalgia.

Of particular import to embedded systems is Moore's law. The exponential increases in device counts and computer performance are expected to continue for another decade and possibly longer [Moore 2003]. Whether another technology will succeed silicon and continue the exponential trend is less clear. Kurzweil [2000] observes that the history of computing demonstrates that when the growth of one technology levels off, another technology emerges to continue the exponential trend. Presently, there are just a few candidates, including nanotechnology and quantum computing, but their prospects are uncertain.

Another important trend is a similar exponential increase in communication system bandwidth, both wired and wireless, sometimes called the Moore's law for communication bandwidth [Cherry 2004].

These trends have two major ramifications. The first is on the kind of systems being constructed. Embedded systems will become even more powerful,

more complex and more interconnected. Their application will expand further, both within existing and to other application areas. Furthermore, as the number and power of embedded computers in an application grows above certain threshold, novel applications may emerge. This point is highlighted in the "Embedded, Everywhere" report [Committee on Networked Systems of Embedded Computers 2001], with the ubiquitous computing as a potential example.

The second is the impact on the engineering of embedded systems. The role of programmability (software) will expand further. Design reuse, be it in the form of SoC platforms, software frameworks, IP blocks or components, and the related design methodologies, will become even more important [Fayad et al. 1999; Keutzer et al. 2002]. The tools will become more powerful and take over an increasing share of engineering activities. Engineering science and design will continue to migrate into these tools, in particular, at the lower levels of abstraction. The level of discourse between the engineer and the computer will continue to rise. The internal operation of the tools will become more involved and their effective use will need an appreciation of their internal operation and their envelope of applicability.

To accommodate these trends in the two-phase approach followed in this paper, the education knowledge areas, their content, and their core hours are adjusted correspondingly.

2.3 Extent and Content of Education Knowledge

Knowledge Area Count. A knowledge area represents a distinct, cohesive body of knowledge. In order for it to be successfully learned and assimilated, it must receive sufficient curricular attention. The limitations on the total curriculum time place a constraint on the number of knowledge areas that can realistically be included. Considering the related computing curricula efforts, the CCCE Body of Knowledge lists 15 knowledge areas in the Engineering Science and Design segment (this does not include professionalism and the two mathematics knowledge areas). The corresponding count in the Computing Curricula for Software Engineering (CCSE) [Joint IEEE-CS/ACM Task Force on Computing Curricula 2004] is similar—14 areas (the Computing Essentials grouping contains 7 distinct areas). This would suggest that, based on the time available and pedagogical considerations, a reasonable number of education knowledge areas is around 15.

Knowledge Areas and Content. A central question is which knowledge areas should be included and what their content should be. To facilitate this task, the paper takes the CCCE body of knowledge as the starting point. In general, there are three main dimensions along which the answer could be structured—implementation technologies, level of abstraction, and application areas.

In the implementation technology dimension, the central technologies for embedded systems are software and digital hardware. The question is whether to include knowledge areas related to other technologies. The "analog electrical" technology is an obvious candidate. There are others. Since many embedded systems interact with their environment through the transfer of force, "analog

mechanical" and possibly even "micromechancial" or "nanotechnology" could be other candidates.

Abstraction is a primary mechanism humans rely on to get around the limitations of their brains. Each knowledge area has a range of abstraction levels associated with it. As the embedded systems become more complex and tools more powerful, abstractions at higher level become more prominent and those at lower levels less so. It is important that the range of levels of abstraction encompassed in the knowledge areas be reasonably large to support the design of present systems and allow generalizations that would facilitate adaptation to new levels as they emerge in the future.

It should be noted that there is more than just one dimension of abstraction. The abstraction chosen depends on the purpose. For example, the commonly mentioned levels of abstraction in embedded systems, e.g., system-algorithm-RTL-gate-circuit [see, e.g., Gajski et al. 2000], are behavioral. Different ranges of abstraction levels arise in nonfunctional considerations, e.g., the performability-availability levels when the extent of the system capability to function is considered. The content of knowledge areas should bring these differences out. After all, a major distinction between an engineer and a technician is in the number of models of the artifact created and the deductions made from them.

The third dimension is that of significant application area(s). Given that one important characteristic of embedded systems is the strength of coupling with the embedding, the curriculum should include one or more prominent application areas in its list of courses. The issue is whether there are some application areas, or classes of areas, that are so central to embedded system engineering that they should be included in its body of education knowledge. There are several candidates including Control and Communications. An important factor in the consideration is whether their concepts, representations, methods, and techniques are, or are likely to be, of major import in the design of embedded systems themselves.

The issue of application areas also arose in the CCSE effort, which requires that the curriculum include exposure, in reasonable depth, to at least one system or application specialty. The report lists 15 possible specialties, but does not rank them. As a general observation, the higher the intended complexity of artifacts engineered by a discipline, the more important the curricular exposure to application domains.[1]

The selection of knowledge areas has an implication on the curricular segments not directly addressed in this paper, namely Mathematics and Basic Sciences. The impact on Mathematics is particularly significant. The separation

[1]A significant and expanding application area with a distinct, separate set of fundamentals may ultimately split off and become another discipline. This was the case with chemical engineering, which originated as a specialty in mechanical engineering for those interested in the engineering of "embedded" mechanical units in chemical plants. In an 1888 description from MIT catalog "[Course X] is arranged to meet the needs of students who desire a general training in mechanical engineering and to devote a portion of their time to the study of the application of chemistry to the arts, especially to those engineering problems which relate to the use and manufacture of chemical products." [Pafko]

between "continuous" mathematics used in the lower levels of abstraction and some application areas, and discrete mathematics needed at higher levels of abstraction, significantly increases the time needs of this segment.

Size of the Core. The decision on how much curricular time the core of each knowledge area should receive is affected by the total time available. Further considerations include the desirability of leaving room for flexibility and innovation, the breadth, the intrinsic complexity, and relative maturity of the discipline, as well as the precedents set by sister disciplines.

The CC efforts use the "lecture hour" as the metric of coverage. At the level of detail of this paper, a coarser metric, the standard course equivalent (SCE) appears more appropriate. One SCE consists of ∼40 lecture hours.

The size of the Engineering Science and Design core in CCCE is 10 SCEs. In CCSE, its size is about the same, but it does not include the curricular time required for coverage of application domains. The 10 SCEs corresponds to one curricular year, slightly less than half of the Engineering Science and Design segment in a typical 4-year engineering program.

These are relatively low figures. They reflect the goals of these efforts, i.e., to define a body of knowledge that should be included in *any* undergraduate Computer/Software Engineering program, regardless of whether it is 3 or 4 years in duration and whether it is administered by CS, ECE, or others.

Considering the level of maturity of the Embedded Systems discipline, its breadth and the typically stricter requirements on ES products that make their design harder, a somewhat larger core size, by 20–30%, would appear appropriate. To that, one should add the time needed for one or two application-related knowledge areas. The total should still leave reasonable curricular flexibility in a 4-year program.[2]

3. COMPUTER ENGINEERING

This section summarizes the Computer Engineering Body of Knowledge presented in the CCCE report [Joint IEEE-CS/ACM Task Force on Computing Curricula 2005]. The report concentrates on the Engineering Science and Design segment of the curriculum. The body pertinent to this segment consists of fifteen knowledge areas. The areas are listed in Table I, together with the minimum core times given in the report and their approximate SCE equivalents.

The table divides the areas into two groups: the major knowledge areas, whose core lecture hour count is ∼0.5 SCE or larger, and the remaining ones. The major areas are roughly ordered by their levels of abstraction.

In addition to the knowledge areas listed in the table, the CCCE report also includes an area called Social and Professional Issues (CE-SPR, 16 core hours). To reflect their importance to computer engineering, the report also details two

[2]What constitutes reasonable must be taken in context. In medicine, a discipline which, in embedded system parlance, is concerned with the operational maintenance of instances of a complex biological system, there is typically little room left beyond the core [see, e.g., School of Medicine Catalog 2002]. Students desiring flexibility must stay around longer.

Table I. CCCE Knowledge Areas and Core Minima (Engineering Science and Design Areas Only)

Label	Major Knowledge Areas	Core Minima	
		Hour	SCE
CE-CSE	Computer System Engineering	18	0.5
CE-ESY	Embedded Systems	20	0.5
CE-NWK	Computer Networks	21	0.5
CE-OPS	Operating Systems	20	0.5
CS-DSP	Digital Signal Processing	17	0.5
CE-ALG	Algorithms and Complexity	30	0.75
CE-PRF	Programming Fundamentals	39	1
CD-CAO	Computer Architecture and Organization	63	1.5
CE-DIG	Digital Logic	57	1.5
CE-CSG	Circuits and Signals	43	1
CE-ELE	Electronics	40	1
	Other Knowledge Areas		
CE-HCI	Human Computer Interactions	8	0.25
CE-SWE	Software Engineering	13	0.25
CE-DBS	Database Systems	5	0
CE-VLS	VLSI Design and Fabrication	10	0.25
		36	0.75

mathematical knowledge areas: Discrete Structures (CE-DSC, 33 core hours) and Probability and Statistics (CE-PRS, 33 core hours).

The report notes that the curriculum must provide adequate coverage of Mathematics and Basic Sciences. For mathematics, besides the two above-mentioned knowledge areas, the report only recommends that the curriculum cover differential and integral calculus and possibly other topics, such as further calculus, differential equations, transform theory, linear algebra, and numerical methods.

The report is less explicit as to what should be covered in the Basic Science segment. It comments that computer engineers need knowledge of basic sciences, such as physics and chemistry, with the former including the basic concepts of electricity and magnetism, but does not detail it.

In terms of implementation technologies, the report includes knowledge areas related to electric analog, digital hardware, and software. The distribution of coverage is biased toward digital. The report does not list any application-related knowledge area nor refer to the need to include them. The coverage of the levels of abstraction peaks at the digital logic-architecture level. The system/software requirement specification level receives fairly small core coverage (about 6 hours total), much less than the circuit and device levels. There is a gap between the programming and the instruction set levels; translation/compilation is only an elective unit in the Software Engineering knowledge area.

One of the knowledge areas is Embedded Systems. It contains mainly a collection of knowledge units addressing incremental knowledge needed to deal with various aspects of computer design under tight cost, latency, power, and reliability constraints (e.g. embedded microcontrollers, RTOS etc.). A more global view is taken in the Computer Systems Engineering knowledge area,

which includes, with roughly equal weight, lifecycle models, requirement analysis and elicitation, specification, architectural design, testing, maintenance, project management, and touches on concurrent hardware/software design.

To assist with curriculum development, an appendix to the report shows several sample Computer Engineering curricula, one for computer engineering programs administered by CS departments, one for programs administered by ECE departments, and one for programs administered by CS departments in conjunction with a department or college of engineering. The curricula meet the minimum core hour requirement and typically include three to four technical electives in the upper years.

4. BODY OF KNOWLEDGE FOR EMBEDDED SYSTEM ENGINEERING

4.1 Education Knowledge

This paper takes a conservative approach to defining the body of education knowledge for embedded system engineering. It takes the CCCE body of knowledge as the starting point and modifies it to reflect the ESE needs. It also adjusts the core time for some areas.

The main changes in the list of knowledge areas include:

1. Redefinition of contents of the Embedded System area and its renaming as Embedded System Design;
2. Addition of two knowledge areas: System Performance and Languages and Translators;
3. Addition of two "dual-use" knowledge areas: Control Systems and Communications;
4. Removal of "small" knowledge areas, with some of their content merged into related larger knowledge areas.

The changes in the core hours include: (a) significant increase for Software Engineering; (b) major increase for Operating Systems, Computer Networks, and Digital Signal Processing; and (c) reduction for Circuits, Digital Logic, and Computer Architecture.

The resulting list of knowledge areas, together with their core allocations, is shown in Table II. Some of the changes are further discussed in Section 4.2. However, space limitations do not allow more detailed description and justification of changes.

4.2 Discussion of Changes

The changes in the SWE knowledge area reflect the fundamental role of software in embedded systems. Software implements the bulk of their functionality and accounts for large fraction of the total ES development budget [Allan et al. 2002]. Its general knowledge base is large and expanding rapidly, as is the body concerned with embedded software [Lee, 2002]. Reflecting that, several core knowledge units were added (embedded software, techniques for reuse, data modeling) and the content and time of some CCCE knowledge units was

Table II. Body of Education Knowledge for Embedded Systems Engineering

	Education Knowledge Area	SCE
ES-CSE	Computer System Engineering	0.5
ES-ESD	Embedded System Design	1
ES-NWK	Computer Networks	0.75
ES-OPS	Operating Systems	1
ES-DSP	Digital Signal Processing	0.75
ES-ALG	Algorithms and Complexity	0.75
ES-PRF	Programming Fundamentals	1.25
ES-CAO	Computer Architecture and Organization	1.25
ES-DIG	Digital Logic	1.25
ES-CSG	Circuits and Signals	0.75
ES-ELE	Electronics	1
ES-SWE	Software Engineering	1
ES-PER	System Performance	1
ES-LTR	Languages and Translation	0.75
ES-CTL	Control Systems	1
ES-COM	Communications	0.75
	TOTAL	14.75

increased (software architectures, software design, software V&V, and software project management).

For the NWK knowledge area, the added knowledge units deal with other OSI layers and non-IP protocols. Content and core time were increased for protocol design and network security. For the OPS area, real-time scheduling and distributed systems knowledge units were added as core and the core hours for device management and security increased. For the DSP area, the additions included deeper coverage of fundamentals (in particular, transforms and convolution) and specific filtering techniques.

Embedded System Design (ESD). This knowledge area replaces the ESY area of the CCCE, most of whose contents were moved to related areas (e.g., RTOS to Operating Systems). The definition of ESD content was based on the view that the central characteristics of embedded systems are the heterogeneity of their implementation technologies (digital hardware, software), the tightness of their nonfunctional requirements, and their complexity. A consequence of the first two is that the parts realized in different implementation technologies often need to be designed and also verified jointly. A consequence of the last characteristic is an increased role for separation of concerns and a bias toward reuse, in the form of hardware/software platforms, software frameworks, IP blocks, and components. A knowledge unit concerned with the theory of hybrid discrete-continuous systems is also included. The area content is shown on the left side of Table III. Note that there is an overlap with the content of advanced Embedded System Design courses such as EECS 249 Design of Embedded Systems [2004].

System Performance. One of the distinguishing characteristics of embedded system is the prominence of their non-functional requirements (how well the system will do what it does). These may be quite tight and include response time, throughput, availability, performability, reliability, and the degree of risk.

Table III. Knowledge Areas Embedded System Design and System Performance

ES-ESD Embedded System Design	ES-PER System Performance
0. History and Overview	0. History and Overview
1. Design Process and Methodologies	1. Performance Metrics
2. Models of Computation	2. Markov Chains, Petri Nets, Que Theo & Nets
3. System Requirement Specification	3. Evaluation: Analytical, Simulation, Tradeoffs
4. Platform/Framework Driven Design	4. Design of Experiments
5. Verification, Validation	5. Response Time, Throughput Modeling
6. Hardware/Software Codesign	6. Availability/Performability Modeling
7. Hardware/Software Coverification	7. Risk/Reliability Modeling
8. Hybrid System Theory	

The tighter the requirements, the greater the design challenges. During design, requirement satisfaction must be checked. There is a rich body of knowledge concerning the derivation of appropriate performance models of computer system designs and their use in the estimation of nonfunctional properties. The related knowledge units are listed on the right side of Table III.

Languages and Translators (LTR). Languages, or, more generally, formal representations of engineering artifacts, are of major importance in embedded system design and maintenance. Their importance begins at requirement specification and extends through all levels of design. They are used to communicate design intent among designers as well as to computer tools. The tools use them to check these representations, to derive their properties, and to communicate design descriptions to other tools.

The tools also translate (synthesize, compile) them to other formal representations. Because of the strictness of nonfunctional requirements of embedded systems, the quality of the translation result is important and the translation process itself may need to be guided. To use these tools effectively, the ESE graduates need an appreciation of their internal operation and their envelope of applicability.

The LTR knowledge area covers the general concepts underlying formal languages, their syntax and semantics, their classification, as well as common families of languages for specification, programming, and hardware description level. It also includes the general principles and major techniques of principal translation tools including programming language and silicon compilers.

Control Systems. Control is a "dual use" knowledge area. One use is application related and concerns the embedded system's control of external world. The notion of external plant whose behavior is characterized by a set of equations is an abstraction of a large number of embeddings that an embedded system may control. The control will be based on feedback and involve minimizing some performance criteria.

The other use is internal. An embedded computer/digital system is a dynamic system in its own. Environmental or system constraints may require that the operation of this internal dynamic system be optimal with respect to some criteria, including responsiveness, throughput, robustness, power consumption, etc. Appropriate "control" mechanisms must be included in the embedded computer

and, in particular, its software, to ensure that its dynamic behavior remains within the required limits.

This is an important point. It has been observed that, while the past of computing was primarily concerned with functionality (i.e., open-loop operation in control system terms), the future will be concerned with optimality (i.e., closed loop) [Spector 2004].This point also motivates major research initiatives, such as the IBM autonomic computing [Kephart and Chess 2003]. These initiatives anticipate that future computing systems will be self-optimizing.

The knowledge of fundamentals and techniques upon which these two uses are based is an important part of ESE body of knowledge. The knowledge units of this area include state variables, feedback, control algorithms, stability, control system design, and others. The mathematical fundamentals underlying this knowledge area are assumed to be primarily covered in the mathematics segment of the curriculum.

Communications. This is again a dual-use knowledge area. Information appliances and telecommunication equipment form a prominent application area for embedded systems. The range of products in this area is expanding rapidly. It is important for an ES engineer to appreciate the issues involved in the communication and processing of information and the constraints involved. Second, the communication channel is a basic abstraction for exchange of information in computer networks. Its characteristics have major influence on the design decisions made, especially if the channel is noisy (e.g., wireless). ESE graduates need deeper understanding of the issues and solutions than what is covered at the level of the NWK area. The Communications knowledge units include fundamentals of information theory, signals and their representations, noise, modulation, and coding.

4.3 Mathematics and Basic Science Knowledge

The above sections deal with the engineering science and engineering design education knowledge. This section briefly comments on the mathematics and basic (natural) science knowledge.

Mathematics. The knowledge areas in Table II must be grounded in sufficient mathematical knowledge. The CCCE report singles out Discrete Structures and Probability and Statistics and gives their knowledge content. These two knowledge areas are also important to Embedded System Engineering and need to be covered. In addition, the mathematical knowledge base for ESE also includes “continuous” mathematics needed for firm grounding of some knowledge areas. Continuous mathematics is particularly required for description and reasoning in the knowledge areas that deal with the analog implementation technology, digital signal processing, the two application-related areas, and system performance. The latter shows that continuous mathematics is also important in describing and reasoning about discrete structures—once the complexity of the structure or of the discrete phenomena occurring in the structure rises above a certain point, continuous approximations offer more tractable solutions.

Basic (Natural) Sciences. Embedded computers do not exist in vacuum; they interact with the world they are embedded in. There are two major kinds of such embedding; the physical world and the information world. Clearly, the curriculum cannot expose the embedded system engineers to all possible applications in these worlds. As an alternative, the curriculum can cover the fundamental concepts and relationships in these worlds. Equipped with such knowledge, ESE graduates should be able to better comprehend what is involved in an application and to communicate with those who specialize in the embedding.

The Communications knowledge area captures such fundamental knowledge about the information world. The Basic Science segment of the curriculum can deliver fundamental knowledge about the physical world. Physics, in particular Mechanics, Electricity and Magnetism, Fluid Mechanics, and Thermodynamics, form the principal sources of this knowledge.

5. SAMPLE CURRICULUM

As in the CC efforts, this section shows a sample implementation of the embedded system engineering curriculum that includes the core of the ES-BEK. This curriculum represents one possible tradeoff between the coverage of education knowledge and the constraints arising from the program length, semester structure, and topic sequencing. Alternative curricula could be devised.

The sample curriculum assumes that the entering students have 1 year of college-preparatory physics and 3 years of college-preparatory mathematics, at the level expected by the Scholastic Aptitude Test. For comparative reasons, the load metric used in the sample CC curricula, i.e., the semester credit hour, is adopted here. This metric takes into account the additional workload due to course laboratories and projects.

5.1 Program Objectives

The program goal is to educate professionals who are able to engineer embedded systems and their software and digital hardware components for a variety of applications. The program graduates will be able to identify engineering issues that arise in the design of such systems and systematically devise effective solutions by applying their knowledge of engineering science and design, and discrete and continuous mathematics. The graduates will be capable of interacting with other professionals who are experts in the application area that the embedded system serves. They will be able to work effectively as team members and understand the impact of engineering solutions in global and societal contexts.

5.2 Curriculum

The curriculum is shown in Table IV. It is intended for institutions that include practical experience (labs/projects) as a part of the related course instead of structuring it as a separate laboratory course. In the load metric used, three credit hours correspond to a semester (13–14 weeks) long course with three 50-min lecture hours and possibly 1 tutorial hour per week. The credit weight of the same course with an additional 3-hour lab every other week, or an equivalent project, is 3.5.

Table IV. Sample Curriculum[a]

Course	Description	Credit	Course	Description	Credit
Semester 1			Semester 2		
MTH 101	Calculus 1	3	MTH 103	Calculus 2	3
MTH 102	Discrete Mathematics	3	MTH 104	Linear Algebra & Numerical Anal	3
PHY 101	Physics 1 (Statics)	3	PHY 102	Physics 2 (E&M)	3
ESE 101	Introduction to ES Engineering	3.5	ESE 103	Computer Organization	3.5
ESE 102	Programming Fundamentals	3.5		Engineering Economics	3
	Total Credit Hours	16		Total Credit Hours	15.5
Semester 3			Semester 4		
MTH 201	Differential Equations	3	MTH 202	Probability and Statistics	3
ESE 201	Algorithms and Data Structures	3.5	ESE 204	Languages and Translators	3.5
ESE 202	Digital Logic	3.5	ESE 205	Computer Systems & Interfacing	3.5
ESE 203	Electrical Circuits	3.5	ESE 206	Signals and Processing	3.5
PHY 201	Physics 3 (Dynamics)	3		Basic Science Elective	3
	Total Credit Hours	16.5		Total Credit Hours	16.5
Semester 5			Semester 6		
ESE 301	Real-Time Operating Systems	3.5	ESE 305	Software Engineering	3.5
ESE 302	Computer Networks	3	ESE 306	Digital System Engineering	3.5
ESE 303	Computer System Performance	3.5	ESE 307	Control Systems	3.5
ESE 304	Electronics	3.5	ESE 308	Communications	3
	Complementary Studies Elective	3		Complementary Studies Elective	3
	Total Credit Hours	16.5		Total Credit Hours	16.5
Semester 7			Semester 8		
ESE 401	Embedded System Design	3.5	ESE 402	Design Project	3
	Technical Elective	3.5		Technical Elective	3.5
	Technical Elective	3		Technical Elective	3
	Basic Science Elective	3		General Elective	3
	Complementary Studies Elective	3		Complementary Studies Elective	3
	Total Credit Hours	16		Total Credit Hours	15.5

[a] Basic Science, General, and Complementary Studies Elective slots may be interchanged.

Curriculum Overview. The first semester of the curriculum contains an "Introduction to Embedded System Engineering" that previews the ESE discipline and introduces professional engineering topics. It has an associated project, in which the students build a small embedded system. The second semester introduces the central component of embedded systems, the computer, at the computer organization, and assembly programming level.

The curriculum then proceeds to build a firm mathematics and physics base in the early years. In parallel, the curriculum begins to develop the engineering science and design foundations, including computer science. Typically, one hardware- and one software-oriented course is included in each semester. With these foundations, the third year includes more system-oriented courses, as well as courses covering the two application-related knowledge areas.

The core part of the curriculum ends with an Embedded System Design course in the seventh semester. This course gives an integrated treatment of the design with the two central technologies, including verification of evolving designs and introduces hybrid system theory.

The curriculum contains a progression of laboratories and projects designed to give broad laboratory experience and exposure to the use of basic tools. The

Table V. Curriculum Summary

Topics	Credit Hours
Mathematics	18
Basic Sciences (core)	9
Basic Sciences (elective)	6
Engineering Science and Design (core)	65
Technical Electives	13
Complementary Studies (core)	3
Complementary Studies (elective)	12
General Electives	3
TOTAL Credit Hours	129

Table VI. Mapping between Knowledge Areas and Courses

Knowledge Area	Course	Knowledge Area	Course
ES-CSE	ESE101	ES-DIG	ESE 202, 306
ES-ESD	ESE 401, 305, 306	ES-CSG	ESE 203, 206
ES-NWK	ESE 302	ES-ELE	ESE 304
ES-OPS	ESE 301	ES-SWE	ESE 305
ES-DSP	ESE 206	ES-PER	ESE 303, 305, 306
ES-ALG	ESE 201	ES-LTR	ESE 204
ES-PRF	ESE 102, 201	ES-CTL	ESE 307
ES-CAO	ESE 103, 205	ES-COM	ESE 308

progression begins with small scheduled laboratories and individual projects in the lower years and evolves to larger, course-related group design projects in the upper years. The last semester includes a major group design project. The project must target an embedded system of appropriate complexity. To fully exploit this opportunity, the students are required to form project groups and define their projects in the preceding semester. Organizationally, this could be done as an add-on to the Embedded System Design course.

5.3 Curriculum Summary

Table V shows the distribution of curricular content, by segment. The distribution meets the accreditation requirement of 0.5 year of Mathematics, 0.5 year of Basic Sciences, 2 years of Engineering Science and Engineering Design, and 0.5 year of Complementary Studies. The curriculum consists of 29 required course and 11 elective courses. Of the 29 required courses, 15 courses have labs or projects. The curriculum incorporates a culminating design experience in the form of a group project.

The primary mappings between the knowledge areas in ESE-BEK and the curriculum courses are shown in Table VI. Social and professional issues are addressed in ESE101 and in complementary studies.

5.4 Observations

This section comments on several points related to the curriculum.

Curriculum Delivery. The outcome of education in complex, interdependent, and evolving disciplines depends, to a great extent, on the delivery of its curricular content. The delivery issue is discussed extensively in general

and engineering education literature, as well as CC reports. The CCSE report guidelines are particularly relevant to ESE, since software engineering shares many characteristics with embedded system engineering. Both are relatively young disciplines concerned with the design of high functionality artifacts.

One of its guidelines, which is of major importance to ESE education, is that of intercourse linkages. The lectures and labs/projects need to highlight the linkages between the courses. In general, these linkages can have horizontal direction (to another implementation technology), vertical (to another level of abstraction), as well as external (to applications or application domains). Also, the role the course content plays in the global ESE picture should be made clearly visible.

Change and Its Dynamics. The ESE discipline is rapidly evolving. It is essential that the students acquire an appreciation of the change, its causes and its dynamics, so that they are better equipped to deal with the changes that will come. Since it is impossible to predict the long-term directions of change, a study of the past can be used as a substitute, as a source of insights and understanding. The lectures in appropriate courses should discuss what affected past evolution of the course matter and extract the underlying patterns. These discussions should expose both the general evolutionary patterns, such as the change of quantity into quality and the thesis–antithesis–synthesis spiral of evolution [Hegel 1812], as well the more specific dynamics of interplay between technology and economics. The students need to see the subject matter as something that evolves and will evolve, not as an inert corpus of knowledge.

Curriculum Breadth. The design of the curriculum intentionally avoided the pattern of providing breadth by mandating a large number of peripherally connected courses. The total program time is fixed; incorporating breadth in this manner comes at the expense of depth. Such exposure is not very effective. The contents of a single peripherally connected course are likely to be forgotten soon unless reinforced elsewhere in the curriculum. Also, should the need to acquire such knowledge arise at some future point in the graduate's career, its limited extent permits its acquisition through self-study in a just-in-time manner. The guiding philosophy in the curriculum design was that of giving depth and narrower, but not too narrow, breadth. The expectation is that, if the base is wide and diverse enough to allow generalizations, whatever broadening may be required later should not be too difficult to acquire.[3]

6. CONCLUDING OBSERVATIONS

The paper presented a curriculum for a 4-year undergraduate program in Embedded System Engineering. The curriculum was developed using a two-step approach similar to the one used by the joint ACM-IEEE task force on computing curricula. A body of education knowledge for Embedded System Engineering was defined first. The body was structured into sixteen knowledge areas. Each area is composed of several knowledge units, some designated as

[3]A somewhat related example is that of learning foreign languages. A bilingual person can learn another language much more easily than an unilingual adult.

core and others as electives. The minimum lecture time for the core of each knowledge area was identified. The CCCE body of knowledge for Computer Engineering was used as the reference in the definition.

The ESE education knowledge served as the base for the development of a sample ESE curriculum presented in Section 5. The curriculum contains solid mathematics and science base, an in-depth exposure to engineering science and design of systems implemented with digital hardware and software, and coverage of two prominent application areas of embedded systems, Control and Communication. The curriculum core takes approximately 3 years of the program and the remaining part is elective.

The engineering science and design of ESE, its implementation technologies, and its applications are evolving rapidly. The curriculum provides a solid foundation to allow the program graduates to adapt to these changes in the near future. Its extent, which includes engineering with several implementation technologies, a range of related levels of abstraction, distinct application areas and substantial mathematics and science base, should facilitate adaptation to the evolution of the ESE discipline and its applications in the longer term.

ACKNOWLEDGMENTS

The author gratefully acknowledges the contributions of his University and industry colleagues to the philosophy, structure and content of the curriculum. Thanks are also due for the inputs and discussions at the author's 2004 sabbatical sites, the School of Computer Science, National University of Singapore, and the Department of Informatics and Applied Mathematics, Federal University of Rio Grande do Norte, Brazil.

REFERENCES

ACCREDITATION BOARD FOR ENGINEERING AND TECHNOLOGY. 2003. *Criteria for Accrediting Engineering Programs (Effective for Evaluations during the 2004–2005 Accreditation Cycle)* (Nov.). http://www.abet.org/criteria.html.

ACM/IEEE-CS JOINT CURRICULUM TASK FORCE. 1990. *Computing Curricula 1991* (Dec.). http://www.acm.org/education/curr91/homepage.html.

ALLAN, A. ET AL. 2002. 2001 technology roadmap for semiconductors. *IEEE Computer* (Jan.). 42–53.

BIGELOW, J. 1865. "An address on the limits of education," read before M.I.T., (Nov.). Available at http://libraries.mit.edu/archives/mithistory/founding.htm.

CANADIAN ENGINEERING ACCREDITATION BOARD. 2003. *Accreditation Criteria and Procedures.* http://www.ccpe.ca/e/files/report_ceab.pdf.

CHERRY, S. 2004. Edholm's Law of Bandwidth. *IEEE Spectrum* (July). 58–60.

CAREER SPACE CONSORTIUM. 2001. Curriculum Development Guidelines (New ICT curricula for the 21st century: Designing tomorrow's education), Office for Official Publications of the European Community, Luxembourg. http://www.career-space.com/cdguide/index.htm.

COMMITTEE ON NETWORKED SYSTEMS OF EMBEDDED COMPUTERS, NATIONAL RESEARCH COUNCIL. 2001. Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers. The National Academic Press. Also available at http://www.nap.edu/books/0309075688/html/.

EECS 249 DESIGN OF EMBEDDED SYSTEMS. 2004. Models, Validation, and Synthesis, University of California at Berkeley. http://www-cad.eecs.berkeley.edu/~polis/class/.

FAYAD, M., SCHMIDT, D., AND JOHNSON, R. 1999. *Building Application Frameworks: Object-Oriented Foundations of Framework Design*, Wiley, New York.

GAJSKI, D. ET AL. 2000. *SpecC: Specification Language and Methodology,* Kluwer Acadmic Publ., Boston, MA.

HEGEL, G. W. F. 1812. *The Science of Logic*. Synopsis in http://encarta.msn.com/encyclopedia_761552560/Hegel.html.

KEPHART, J. O. AND CHESS, D. M. 2003. The vision of autonomic computing. *IEEE Computer*, (Jan.), 41–50. See also *Autonomic Computing Manifesto*, IBM, http://researchweb.watson.ibm.com/autonomic/manifesto/autonomic_computing.pdf.

KEUTZER, K. ET AL. 2002. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Trans. Computer Aided Design*. (Dec.). 1523–1542.

KURZWEIL, R. 2000. *The Age of Spiritual Machines: When Computers Exceed Human Intelligence*, Penguin, New York.

LEE, E. A. 2002. Embedded software. In M.Zelkowitz (ed.), *Advances in Computers*, Vol. 56. Academic Press, London.

MOORE, G. E. 2003. No exponential is forever: But "forever" can be delayed! Keynote address. In *Proceedings of the IEEE International Solid-State Circuits Conference,* Section 1.1.

NELSON, V. ET AL. 2004. Computing curriculum—Computer engineering (CCCE) "A model for computer engineering curricula in the next decade," presentation at American Society for Engineering Education Annual Conference and Exposition (ASEE'04), (June).

PAFKO, W. History of chemical engineering & chemical technology. http://www.pafko. com/history/.

Scholastic Aptitude Test, Subject Tests: Mathematics, Chemistry, Biology and Physics. http://www.collegeboard.com/prod_downloads/sat/SAT2_Math1and2_Bio_Ch_Ph.pdf.

School of Medicine Catalog 2002–03, Stanford University, Stanford, CA, 2002.

SPECTOR, A. Z. 2004. IBM Research Division: Presentation to Snowbird, presented at Computing Research Conference at Snowbird, (July). http://www.cra.org/Activities/snowbird/2004/slides/spector.ibm.pdf.

THE JOINT IEEE-CS/ACM TASK FORCE ON COMPUTING CURRICULA. *Computing Curricula 2001—Computer Science,* Final Report, Dec 2001. http://www.computer.org/education/cc2001/final/cc2001.pdf.

THE JOINT IEEE-CS/ACM TASK FORCE ON COMPUTING CURRICULA. *Computing Curricula: Software Engineering*, Final Report May 2004. http://sites.computer.org/ccse/volume/FinalReport-5-21-04.pdf.

THE JOINT TASK FORCE FOR COMPUTING CURRICULA. 2004. *Computing Curricula 2004: Overview Report including A Guide to Undergraduate Degree Programs in Computing for Undergraduate Degree Programs in Computer Engineering, Computer Science, Information Systems, Information Technology, Software Engineering*, A cooperative project of ACM, AIS, and IEEE-CS (Nov.). http://www.acm.org/education/Overview_Draft_11-22-04.pdf.

THE JOINT IEEE-CS/ACM TASK FORCE ON COMPUTING CURRICULA 2005. *Computing Curricula: Computer Engineering*, Ironman Draft. (June). http://www.eng.auburn.edu/ece/CCCE/CCCE-IronDraft-2004June8.pdf.

University of Waterloo Undergraduate Calendar 2004–05, University of Waterloo, 2004, http://www.adm.uwaterloo.ca/infoucal/INTRO/prog_avail.html.

Received September 2004; revised February 2005; accepted May 2005