Editorial

As embedded systems activities are increasingly being recognized as a single coherent endeavor, it is not surprising that those concerned with education are striving to define appropriate curricula for this emerging engineering discipline. In this special issue of the Transactions on Embedded Systems, we focus on university education. We provide examples of existing practices, look at initiatives to develop new curricula, and take an educationally centered view on the evolution of this new engineering domain. Although there is a significant and growing agreement as to what defines embedded systems engineering, there is not a universal consensus as to what constitutes the discipline boundaries and, hence, what should be covered in “university provision.” This special issue is, therefore, of interest to those involved in education and those interested in the development of the discipline itself. Industrial practice constrains this development, but so does the knowledge base and skills that graduates obtain. Indeed, it is perhaps only possible within universities to take a holistic view of any engineering discipline to define its core topics, necessary foundations, and linked themes.

In this special issue eight papers are presented. Four describe experiences gained from teaching (partially or completely) embedded systems as a distinct subject and are from United States institutions that are well known for their research work in embedded systems.¹ What is of interest here is not only what is taught, but how the required teaching is organized. Important issues are the balance between formal lectures and project work, between student-centered and class-based learning, between theory and practice, and between methods and tools. The first paper describes the courses in embedded systems at the University of California at Berkeley with particular emphasis on the graduate curriculum. Their guiding principle is to bring together system theory and computer science. The curriculum has been growing since 1988 out of a bottom-up approach typical of U.S. institutions and is spread over a number of experimental and established courses that provide the foundation from which a future graduate and undergraduate program in embedded systems will rise as a coherent whole. At the CMU, an undergraduate course has evolved over the last three decades. Key areas covered are small and single microprocessor applications, control systems, distributed embedded control, system on chip, networking, embedded PCs, critical systems, robotics, computer peripherals, wireless data systems, signal processing, and command and control. Additional cross-cutting

¹Since the readers of the special issues are bound to have different experience with learning institutions, we recall that in the United States courses are term-based, i.e., they are taught over 10 weeks for quarter-based schools or over 15 weeks for semester-based schools. The number of units that a course carries is equal to the number of contact hours plus one-half of the discussion session contact hours; for graduate classes, discussion sessions are rare. The number of units also reflects the amount of effort a student has to put in the course material to pass the exam. Exams are delivered only once at the end of the term. GPA (grade-point average) is computed by multiplying the grade obtained in a class by its number of units, summing this quantity over all courses taken, and dividing by the sum of the total number of units.

areas cover security, dependability, energy-aware computing, software/systems engineering, real-time computing, and human–computer interaction. The paper from UCLA reviews their curriculum, but focuses more on the importance and organization of student projects. The final paper in this set describes the efforts conducted at Vanderbilt University to establish a curriculum in embedded software and systems. Given the compartmentalized nature of traditional engineering schools, where each discipline has an independent program of study, they devised innovative ways to bring together the two disciplines. The emphasis of the paper is on using learning technology to construct, manage, and deliver computer-aided learning modules that can supplement the traditional course structure in the individual disciplines through out-of-class and in-class use.

The next three papers aim at presenting the criteria to follow for developing modern curricula for embedded systems. The first paper is a personal view by Rudolph Seviara. He builds from the IEEE-CS/ACM Computer Engineering curriculum to detail a complete undergraduate degree course. Particular emphasis is placed on the role of control theory, both as an application area and as a key component of the internal management of any dynamic embedded system. In contrast to this personal view, the next paper presents guidelines distilled from a large consortium (the ARTIST group) of mainly, but not exclusively, European academics. This paper has a software focus and identifies six major bodies of knowledge: formulations, basic control and signal processing, theory of computing, real-time computing, distributed computing, and evaluation and optimization (of extra-functional properties). To these they add two transversal themes: system architecture and design and applications. The third paper in this set is again very different in its pedigree. It describes a national program organized by the Taiwan government to significantly increase the quality and quantity of graduates specializing in embedded software. The program is top-down—courses are designed and prototyped before being made available (with full teaching support materials) to a wide range of universities. As well as courses, emphasis is placed on the student experience, with a number of joint activities (including competitions) being central to the program. International cooperation is also a key part of their plans.

Taking these seven papers together (four on experience and three on future plans), a perhaps surprisingly consistent view is provided of what a student with a degree in “Embedded Systems Engineering” needs to know. It is clear that this is not just computer science with a few extra courses, or electronics with software engineering on top, but an integration of material whose formulations need to be comprehensive, but coherent. For example, to deal with discrete and continuous time and to deliver effective abstractions without ignoring critical nonfunctional properties, such as power, communication and computational resources, security, reliability, and safety. The final paper in this special edition explores this narrative further by applying didactical analysis to embedded systems education. Didactics is the study of subjects as they are constructed and taught within educational establishments and institutions. It provides four dimensions on which to judge the current state of a body of knowledge that is to be incorporated into a single coherent subject. The paper notes that embedded

systems engineering is progressing down a path that has been trodden previously by subjects such as chemical engineering and mechatronics. The starting point is a collection of distinct disciplines that are linked together in educational programs, then cross-disciplinary new courses are developed, and, from these, themes emerge which give rise to new curricula. The significance of the original disciplines diminishes as a distinct new discipline emerges. This will be supported by new organizations, such as, departments, schools or faculties. This progression is illustrative, but it also has an important resonance for how we teach the subject. While we are in the thematic stage (which seems to be the case) educationalists recommend certain “rules” for constructing the curriculum. The final paper by Grimheden and Torngren covers this material and contributes an excellent conclusion to this special issue.

ALAN BURNS

ALBERTO SANGIOVANNI-VINCENTELLI