Editors: Jerzy W. Rozenblit, University of Arizona, ECE 320E,Tucson, AZ 85721; jr@ece.arizona.edu; and Sanjaya Kumar, Honeywell Technology Center, MS MN65-2200, 3660 Technology Dr., Minneapolis, Minnesota 55418; skumar@htc.honeywell.com

Toward Synergistic Engineering of Computer Systems

Jerzy W. Rozenblit, University of Arizona
Sanjaya Kumar, Honeywell Technology Center

Computer-based systems—systems whose behavior is primarily computer-controlled—are everywhere. They run the gamut from low-complexity, embedded cruise-control units to the very complex avionics systems. Yet these heterogeneous systems have some characteristics in common: They combine application-specific and off-the-shelf software, hardware, and interface elements, each of which can have varying degrees of structural, functional, and behavioral complexity.

Given the tendency of computer-based systems to grow ever more complex, there is a recognized need for new ways to manage that complexity.

Traditional engineering practice, which has evolved through experience, is to develop a system's architecture, hardware, and software as separate entities. But this approach has led to cost and schedule overruns, as well as to systems that do not perform as intended. Today it is increasingly important that a system's hardware, software, interfaces, and related functions be developed *synergistically*—with an understanding that every system element will influence other elements. It is this need for a systematic, integrated engineering approach that caused us to accept the challenge of developing this department in *Computer*.

Our objective is to provide a forum for sharing problems, experiences, and solutions that can help establish bridges among hardware engineers, software engineers, systems engineers, and project managers. To this end, we plan to present articles encompassing a broad spectrum of applications (both *large* and *small*) and application domains. We will seek to publicize innovative solutions and engineering methodologies that address

- complexity,
- model-based engineering (as exemplified by hardware/software codesign), and
- process management.

We believe these three areas are critical to advancing the successful, integrated engineering of computer-based systems.

COMPLEXITY

Complex systems design—formulated as the process of translating requirements into an actual product—belongs to the class of computationally hard problems. It is highly unlikely we will ever devise an algorithm that can design a complex system better than a human can. Whereas subsets of the design process—layout, routing, and interface checking, for example—can be done by electronic design automation tools, no design tool could automatically create an entire avionics system or a massively parallel computer.

System design and development is both an art and science—it requires original solutions, not fixed algorithms. Key to the process are individual creativity and the perception of customer needs. Thus we should focus on the "human in the loop" when addressing the complexity of both the engineering process and its product.

Clearly, products are created by a process. Yet process is often driven by product—by a product's underlying structure and functionality. This sometimes ambiguous relationship between process and product requires further research. We must continue to explore knowledge-representation and process-management techniques.

The tripartite representation of systems through object, functional, and behavioral models is becoming a de facto standard in the complex-systems design community. Similar strides are being made to develop reliable process-modeling techniques.

System design and development is both an art and science—it requires original solutions, not fixed algorithms.

However, we still lack an operational means of handling the combinatorial complexity of the underlying design space and its search processes. We may accomplish this in the future with better modeling and simulation-based techniques.

MODEL-BASED ENGINEERING

Current practice dictates the separation of the hardware and software development paths early in the design cycle, with very little interaction between them until system integration. Because integration occurs late in the process, any problems encountered are costly. For example, the premature selection of unacceptably slow processors may require that the software attempt to correct the inadequacy. Conversely, poor software performance may necessitate the development of special-purpose hardware.

Hardware/software codesign attempts to integrate the hardware and software development paths, providing a more synergistic approach to system design. Using a codesign environment, engineers can determine the best mixture of hardware and software within a particular system.

Ideally, an integrated codesign environment supports hardware/software evaluation throughout the development process using models with various levels of detail. Appropriate abstractions are crucial. Abstractions let us focus on aspects of interest and thus manage complexity. More detailed descriptions of critical portions of the system let us assess risk. Using models, we can analyze the consequences of different design decisions within the system as a whole, gradually refining the system description into a hardware/software implementation—an idea called *model continuity*. Model continuity allows for the continuous and incremental evaluation of the system. Important issues include determining the amount of detail required within a model to obtain "reasonable" results and supporting hardware/software modeling at different levels of detail.

We advocate tools and techniques that foster the integration of the hardware and software perspectives. An important means of achieving this integration is through unified representations. Unified representations can be used to model a system independent of its implementation in hardware or software, or to model both hardware and software for combined evaluation. A number of modeling representations and formalisms have been proposed in the literature and applied to computer-based systems. They include dataflow diagrams, finite state machines, Petri nets, specialized algebras, and object-oriented representations.

Our position is that these formal specification techniques have limited utility without a systematic modeling methodology to guide their use. We will seek to present such modeling methodologies in this department, focusing on integrated design environments that use unified representations.

PROCESS MANAGEMENT

It is very important to capture the engineering process, its formalization, and its management. Well-structured domains such as chemical and civil engineering have established engineering foundations and principles. But even in these disciplines, methodological problems and errors can occur as processes and projects become more complex. They may stem from attempts to solve imprecisely stated problems or from improper or inadequate use of existing technologies. Most often, however, they result from the lack of an underlying engineering process methodology and personnel specifically trained to manage this process.

This problem is especially acute in the engineering of complex computer-based systems. Factors contributing to the complexity of the engineering process are the multitude of different principles, techniques, methods, and tools employed by designers and engineers, and the relentless introduction of new technologies that must be harnessed to support the management of such processes—for example, high-speed computer networks, standard data exchange formats, and multimedia.

However, these new technologies can facilitate a positive change in the work style and team approaches, as demonstrated by the new paradigms of collaborative, distributed design techniques. We will seek to show how new approaches can harness the available technology to handle the complexity of large-scale projects; to better support concurrent, collaborative, and distributed development of systems; and to automate or partly automate routine engineering activities. We will go beyond traditional approaches stemming from the operations research and computer science disciplines that represent, plan, and optimize project engineering. We will focus on a new paradigm, *process modeling*, and examine how it can be extended from the software domain to integrated engineering.

We feel that process modeling

- adequately captures multiperson design activities with a high degree of cooperation, distribution, and coordination of various tasks;
- reflects a computer-based systems development process that is evolutionary in nature; and
- provides a means to rapidly accommodate new design paradigms as new technologies emerge.

In the next five columns, we will be looking at these issues in greater detail. We will seek representation from researchers and practitioners from a broad spectrum who possess the expertise needed to build complex heterogeneous systems. We welcome your participation. ❖

Jerzy W. Rozenblit is an associate professor in the Department of Electrical and Computer Engineering at the University of Arizona.

Sanjaya Kumar is a senior research scientist at the Honeywell Technology Center.

For More Information

Blanchard, B., *Engineering Organization and Management*, Prentice Hall, Old Tappen, N.J., 1976.

Chapman, W.L., J.W. Rozenblit, and T. Bahill, "Complexity of the System Design Problem," *Proc. 1995 IEEE Symp. and Workshop Systems Engineering of Computer Based Systems*, CS Press, Los Alamitos, Calif., 1995, pp. 51-57.

Curtis, B., M. Kellner, and J. Over, "Process Modeling," *Comm. ACM*, 35(9), 1992, pp. 75-90.

Franke, D.W., and M.K. Purvis, "Hardware/Software Codesign: A Perspective," *Proc. 13th Int'l Conf. Software Eng.*, IEEE CS Press, Los Alamitos, Calif, 1991, pp. 344-352.

Kumar, S., et al., *The Codesign of Embedded Systems: A Unified Hardware/Software Representation*, Kluwer Academic Publishers, Boston, 1996.

Rozenblit, J.W., and K. Buchenrieder, *Codesign: Computer-Aided Software/ Hardware Engineering*, IEEE Press, Piscataway, N.J., 1994.

Wymore, W.A., *Model-Based Systems Engineering*, CRC Press, Boca Raton, Fla., 1993.