Looking Back, Moving Forward

Jerzy W. Rozenblit, The University of Arizona
Sanjaya Kumar, Motorola

This issue marks the end of the Integrated Engineering column, which first appeared in *Computer* in 1997. From the outset, we sought to provide a forum for researchers and practitioners to share thoughts and experiences on synergistic engineering of complex computer-based systems, which span a variety of application domains such as automobiles, avionics, and telecommunications. We now look back at what we accomplished and reflect on what the future holds.

SYNERGISTIC ENGINEERING REVISITED

Our first article ("Toward Synergistic Engineering of Computer Systems," Feb. 1997, pp. 126-127) outlined the belief that integrated computer systems require a mixture of general-purpose and application-specific hardware and software to successfully perform their intended function, while balancing several attributes of interest including performance, cost, safety, and availability.

Several common themes emerge from the design of these systems, particularly the need to support

- complexity management,
- cooperative hardware and software development, and
- process management.

In this context, process management encompasses the multiperson cooperation, distribution, and coordination of tasks essential to system development.

Using these themes as a guideline, we solicited a wide range of articles that emphasized the synergistic activities necessary to address various aspects of system development. We have tried to blend coverage of state-of-the-art technology with discussions of future research opportunities, emphasizing where possible the application and benefit of techniques to real-world problems and situations.

A synergistic approach to developing complex computer-based systems will continue to play an important role.

As a sample of our column's scope, we have included manuscripts representative of work in theory-based, systems-level methods; hardware, networking, and configurable computing; and applications. Specifically, these manuscripts addressed

- integrated hardware-and-software device technologies, including tools, that support microelectrical-mechanical systems and systems-on-chip;
- hardware-software codesign, including smart cards, networked embedded systems, configurable computing, sensing networks, and cosimulation environments;
- software technologies such as middleware that support adaptive quality of service, reuse, and object-oriented approaches;
- fundamental methodologies including axiomatic design, model-integrated computing, and systems ecology; and
- applications such as avionics systems, integrated computer-aided manufacturing solutions, and distributed systems control.

These columns provide a testimony to the creativity of engineers and scholars who advance the field as new needs arise.

FUTURE DIRECTIONS

Although it would be presumptuous to predict what specific techniques and technologies will emerge to support the synergistic engineering of computer systems, we can outline issues that, however intractable, pose interesting research and development challenges.

Computer's November 2001 edition, focusing on the engineering of computer-based systems, addresses some of these challenges with articles that describe

- model-based engineering for virtual prototyping of complex systems through incremental evaluation of design solutions;
- the need for an integrated architecture that synergistically combines the hardware, software, and communication parts of the system under design; and
- system and tool composition technologies that form the basis for building integrated-development environments.

In our view, designers' continued inability to gradually refine the systems-level model into hardware-and-software realizations transcends all these issues. Effectively solving the problem of model continuity is a sine qua non of successful design automation at a high level of system specification abstractions.

To accomplish high-level design auto-

mation, which should provide faster time to market and lower design cost, specification models must

- exhibit high levels of generality—be sufficiently generic to represent a general model of computation exhibited by heterogeneous systems, including complex timing interdependencies;
- support representation at various abstraction levels;
- foster modular and hierarchical construction; and
- support consistency and completeness checks throughout the range of abstraction levels.

The availability of such models will ultimately lead to *model compilation*. Just as we can now compile software or hardware languages, we must develop techniques for compiling system specification languages into integrated "executable" physical realizations.

Future integrated-specification models must also support adaptability. Designs are now emerging, especially in embedded systems. Conceptually, model-based techniques offer great potential for designing such systems because they facilitate implementation-independent specifications.

One desirable research goal is the ability to map variable model structures onto reconfigurable platforms while preserving behavioral characteristics, including timing constraints that are typically verifiable at the realization but not the model-specification level. Extending modeling techniques to capture real—as opposed to simulation—time constraints presents another challenge.

Along with developing integrative system model specifications, we need methods for design-model testing and requirements-compliance verification. In place of the largely ad hoc practices that industry currently uses, we must define a process that inputs system requirements captured in natural-language format and outputs a minimal set of test scenarios that cover all requirements.

To be comprehensive, the test cases must cause at least one state transition—possibly generating a system response—associated with each uniquely identified system requirement. Researchers have achieved some progress in automating this process, but much work remains before we can include temporal constraints in both formal requirements specification languages and testing methods.

ACKNOWLEDGMENTS

The contributions that appeared in this column would not have been possible without the support of many individuals. We thank the contributors for their patience and cooperation during the editorial process. For those whose contributions we could not, regretfully, include, we appreciate your participation and encourage you to pursue other opportunities in *Computer*.

We also thank our readers for their numerous comments on various columns—we greatly valued your feedback and participation. In addition, our participation as coeditors would not have been possible without the support of our colleagues at the University of Arizona, Honeywell Laboratories, and Motorola.

Finally, we thank those at *Computer* who worked with us to produce high-quality articles during the past five years. Editor in Chief Jim Aylor provided the spark that established the Integrated Engineering column, while the staff's strong attention to detail and commitment to producing engaging copy proved integral to the column's success.

We believe that a synergistic approach to developing complex computer-based systems will continue to play an important role in a broad range of future military and commercial systems. Further, we expect that the Embedded Computing column will expand *Computer*'s coverage of integrated engineering topics to encompass new and exciting uses for the dynamic fusion of hardware and software components.

Jerzy W. Rozenblit *is a professor of electrical and computing engineering at the University of Arizona. Contact him at jr@ece.arizona.edu.*

Sanjaya Kumar *is a principal staff engineer at Motorola. Contact him at sanjaya.kumar@motorola.com.*