University of Pennsylvania
ScholarlyCommons

Departmental Papers (CIS) | Department of Computer & Information Science

November 2005

Opportunities and obligations for physical computing systems

John A. Stankovic
University of Virginia

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

Aloysius Mok
University of Texas at Austin

Raj Rajkumar
Carnegie Mellon University

Follow this and additional works at: https://repository.upenn.edu/cis_papers

Recommended Citation

John A. Stankovic, Insup Lee, Aloysius Mok, and Raj Rajkumar, "Opportunities and obligations for physical computing systems", . November 2005.

Copyright 2005 IEEE. Reprinted from *Computer*, Volume 38, Issue 11, November 2005, pages 23-31.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Pennsylvania's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/cis_papers/222
For more information, please contact repository@pobox.upenn.edu.

Opportunities and obligations for physical computing systems

Abstract

The recent confluence of embedded and real-time systems with wireless, sensor, and networking technologies is creating a nascent infrastructure for a technical, economic, and social revolution. Based on the seamless integration of computing with the physical world via sensors and actuators, this revolution will accrue many benefits. Potentially, its impact could be similar to that of the current Internet. We believe developers must focus on the physical, real-time, and embedded aspects of pervasive computing. We refer to this domain as physical computing systems. For pervasive computing to achieve its promise, developers must create not only high-level system software and application solutions, but also low-level embedded systems solutions. To better understand physical computing's advantages, we consider three application areas: assisted living, emergency response systems for natural or man-made disasters, and protecting critical infrastructures at the national level.

Comments

Copyright 2005 IEEE. Reprinted from *Computer*, Volume 38, Issue 11, November 2005, pages 23-31.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Pennsylvania's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

This journal article is available at ScholarlyCommons: https://repository.upenn.edu/cis_papers/222

Opportunities and Obligations for Physical Computing Systems

John A. Stankovic
University of Virginia
Insup Lee
University of Pennsylvania
Aloysius Mok
University of Texas at Austin
Raj Rajkumar
Carnegie Mellon University

Seamlessly integrating computing with the physical world via sensors and actuators, physical computing systems promise to give society an improved living standard, greater security, and unparalleled convenience and efficiency.

The recent confluence of embedded and real-time systems with wireless, sensor, and networking technologies is creating a nascent infrastructure for a technical, economic, and social revolution. Based on the seamless integration of computing with the physical world via sensors and actuators, this revolution will accrue many benefits. Potentially, its impact could be similar to that of the current Internet.

We envision data and services that will be available any place, any time, to all people, not just technically sophisticated organizations and individuals. Major systems such as those in transportation, manufacturing, infrastructure protection, process control, and electricity distribution networks will become more efficient and capable. People will be safer and experience an improved standard of living. New applications not even imagined today will become a reality.

Although ingredients of this vision have existed for several years and the concept of pervasive computing does not differ radically from the work we describe, we believe developers must focus on the physical, real-time, and embedded aspects of pervasive computing. We refer to this domain as *physical computing systems*. For pervasive computing to achieve its promise, developers must create not only high-level system software and application solutions, but also low-level embedded systems solutions.

To better understand physical computing's advantages, we consider three application areas: assisted living, emergency response systems for natural or man-made disasters, and protecting critical infrastructures at the national level.

SISAL

Projections of world population indicate that many countries, especially in the West, face a decline in population and in the home care and health workforce. There is also a simultaneous and unprecedented global surge in

0018-9162/05/$20.00 © 2005 IEEE Published by the IEEE Computer Society

Devices and appliances for assisted living must be easy to use and highly dependable.

the percentage of older people. Many senior citizens will live alone in the future. If they fall or become sick, the ability to monitor their well-being remotely will not only save lives but also provide peace of mind for their loved ones. Even under nonemergency conditions, their living quality would improve significantly if their food consumption and medication could be monitored and adjusted automatically. Groceries and health supplies could be automatically tracked and replenished as well. *Sensor information systems for assisted living* (SISAL) specifically target this growing need.

Such systems will likely have as significant an effect on societal well-being as any application of embedded, real-time, wireless-network, sensor-information-system, and e-commerce technologies. Broadly, SISAL encompasses intelligent monitors, smart devices, autonomous appliances, sensor networks, and information systems designed to help elderly and disabled individuals live independently, enhance their safety and quality of life, ease the burden of their caretakers, and reduce home care costs. Low-cost and dependable, SISAL will not only help decrease care costs for the elderly and disabled, but will also benefit people of all ages directly, much as ramps for handicap access in airports now let all travelers pull their carry-ons effortlessly.

Devices and appliances for assisted living must be easy to use and highly dependable. A majority of them will be mission- and safety-critical. Ideally, they should have self-diagnostic and -healing capabilities. SISAL must be affordable by people of all income levels. Further, users who have little or no technical skills, knowledge, or discipline must be able to maintain and upgrade these systems.

Many SISAL components cannot be effective as stand-alone devices, but must be integrated seamlessly into a ubiquitous system. Typical SISALs can contain a few thousand to tens of millions of devices and appliances. SISAL components will likely vary widely in functionality, size, complexity, and cost. They will likely be produced by hundreds or even thousands of companies, big and small, worldwide.

SISAL component examples include the following:

- smart prescription dispensers that a caregiver can program, monitor, and validate remotely;
- heartbeat monitors that can record heartbeat samples, detect irregular patterns, and send notifications or alarms;
- robotic helpers designed to enhance accessibility, mobility, and safety; and
- smart pantries that inventory grocery and food items and notify designated suppliers for just-in-time replenishments.

Although the life cycles of some SISAL components could stretch as long as 10 to 20 years, the computing and communication platforms comprising their support infrastructure should be able to evolve more rapidly. Further, given the mission-critical nature of their role, there must be no glitches that interrupt service to the user while SISAL components and infrastructure evolve independently. Building systems with these characteristics and required qualities poses multiple research and practical challenges.

EMERGENCY RESPONSE SYSTEMS

Physical computing systems in the form of wireless sensor networks have great potential to enable fast response to emergency situations such as earthquakes, chemical spills, forest fires, and terrorist attacks. Responding effectively to such emergencies would require rapidly deploying thousands of sensor nodes in the region, possibly from aircraft. Each node must be cheap and, therefore, will likely have limited capacity. However, the nodes could collectively assess the situation, detect the presence of possible survivors, send vital information to the nearest rescue teams, identify a safe exit path, and monitor any subsequent events of interest in the area. Because such emergencies can last for several hours to days and are highly dynamic, the system must adapt to the changing environment, be robust, utilize the limited available resources efficiently, and meet timing requirements.

While these systems have great potential, many technical challenges must be addressed before they can be implemented. For example, because sensor networks deal with real-world situations, their data services must meet timing constraints. Given sensor networks' limited resources and unpredictability, guaranteeing hard real-time constraints is infeasible. Solutions that provide certain probabilistic guarantees for timing constraints are more practical.

Novel approaches to constructing sensor networks with event services for emergency response are also necessary. These services must consider several fundamental requirements, including efficient energy consumption and robust and timely delivery of data and events. Ultimately, these systems will provide a facility that lets the application specify

the desirable data and events, then deliver them in a timely manner.

PROTECTING CRITICAL INFRASTRUCTURES

Protecting and improving critical infrastructures require long-range research in physical systems technology. We need new concepts for implementing robust and secure infrastructures for complex networks of highly interactive physical systems such as the power grid, telecommunications, banking, and transportation. These often dynamically reconfigured networks must deal with the physical world.

Current infrastructures are vulnerable because traditional *digital control systems* and *supervisory control and data acquisition systems* have not been seamlessly integrated with computer networking, information technology, computer security, and other key aspects of physical computing systems.

These infrastructure systems are increasingly interdependent. For example, an air traffic control system could be challenged by physical or cyberspace attacks. Millions of people fly every day. About 500 FAA-managed ATC towers, along with 180 low-altitude radar control systems and 20 enroute centers, control the US's air space and its 10,000 or so airports.

All these locations depend on power. Attacks on the power grid could affect air traffic control, telecommunications, and all sectors that depend on power. Solutions developed in physical-computing-systems research could be applied to critical infrastructures to make them reliable, safe, and more efficient. For example, security and isolation solutions could decouple critical aspects of these systems from easy external access. Physical security could be enhanced by surrounding the physical infrastructure with a wireless sensor network.

RESEARCH QUESTIONS AND ISSUES

The fundamental research question developers must answer involves understanding how to cost-effectively develop reliable large-scale *real-time embedded systems*. Building these RTESs requires a deep understanding of how to model and analyze large-scale systems' aggregate behaviors, which necessitates large-scale coordination and cooperation. Such understanding is necessary to predict and control system behavior and robustness and to efficiently build and maintain such desirable traits as ease of composition, analysis, testing, diagnostics, quality of service, and usability properties. We classify this work into the following categories: theory, modeling, and analysis; programming abstractions; embedded systems software and hardware; system integration; security and privacy; and validation and certification.

Theory, modeling, and analysis

Traditionally, practicing engineers and programmers have taken an ad hoc approach to creating embedded software systems. The domain-expert designer alone retains knowledge of the system's behavior and functionality, which can only be imperfectly captured and translated into system products by the development engineer and programmer. Design based on mathematical modeling plays a critical role in other engineering disciplines, such as structural engineering. The complexity of embedded software systems has increased to the point where envisioning the development of high-quality embedded software systems without using such techniques has become impossible.

Solutions developed in physical-computing-systems research could be applied to critical infrastructures to make them reliable, safe, and more efficient.

Developers need new theoretical and analytical techniques to deal with aspects that have not been considered traditionally. For example, basic results exist on capacity bounds for wireless sensor networks. This important work is based on severe assumptions, however. New throughput and capacity bounds that relax assumptions and match real-world systems would be extremely valuable. One possibility, the use of hybrid system models,[1] would let developers model both discrete and continuous behaviors to describe dynamic aspects of wireless sensor networks' environmental effects and thus relax static assumptions.

Refined analysis. Given that they are in an early development stage, few analytical results have been recorded for these networks so far. Researchers are busy inventing new protocols and applications for wireless sensor networks. They build solutions, then test and evaluate them through simulations or testbeds. Occasionally, developers deploy an actual system. Thus, some empirical evidence has begun to accumulate. However, the field requires a more scientific approach in which researchers can design and analyze a system before deployment. The analysis must accurately reflect the system's efficiency and performance and determine that the system will meet its requirements. To achieve this, researchers must determine the

- node density required to meet the system's lifetime requirements;
- sensing and communication ranges needed to detect, classify, and report a target to a base station by a deadline;

Large-scale optimization of quality-of-service parameters and resource allocation must be carried out to maximize the functioning components' benefits.

- sensing range and percentage of awake nodes needed to guarantee a certain degree of sensing coverage for a system;
- possibility of designing different and better wireless sensor network topologies to monitor the health and safety of physical infrastructures such as buildings, bridges and campuses; and
- assurance that all traffic will meet its deadlines given n streams of periodic sensing traffic characterized by a start time, period, message size, deadline, source location, and destination location for a given network.

To resolve this final requirement, researchers must take into account the interference patterns of wireless communication. Once they develop analysis techniques and solutions for these types of questions, they must validate them with real systems.

Group management. A topic well-studied in distributed computing, group management provides a set of definitions, an associated theory as a function of different fault models, and many protocols that meet the various semantics associated with the theory. Researchers now require a new and similar theory for group management in wireless sensor networks. This theory would center on a new set of fundamental premises such as

- it is not reasonable to know all members of a group,
- approximation is the best accuracy level that can be achieved, and
- protocols must scale to large systems composed of thousands of nodes.

Dealing effectively with large-scale RTESs requires applying the notion of composition. The models used should be composable to facilitate reuse and sharing and to allow the construction of a complex system that uses simpler components. The composition can be for either homogeneous or heterogeneous models.

Traditionally, research on formal methods has concentrated on composing specifications in the same modeling language or paradigm. To elevate this modeling, researchers must investigate how to compose models with different purposes—such as choosing one design for the physical layout of a sensor network and another for the protocol used between sensor nodes—to determine interaction between different views and to understand the overall design.

Partial and local failures in large-scale physical computing systems will probably be unavoidable. The challenge at the system level will be to contain any such failures within a well-known perimeter and dynamically reconfigure the system to bypass the affected region. Large-scale optimization of quality-of-service parameters and resource allocation must be carried out to maximize the functioning components' benefits. The system must be able to make such decisions even under adverse environmental factors, such as noise and hostile agents, that operate beyond the system's direct control.

Automating design. In the 1980s and 1990s, design automation tools enabled the semiconductor revolution by separating design and test engineering from fabrication and by significantly improving design productivity. A similar revolution must occur in the design and development of physical computing systems for our vision to become a reality.

The functional behaviors of software components must be separated from the system's parafunctional needs for timeliness, dependability, quality of service, and modalities.[2] Embedded software components must be designed and reused at an abstraction level independent of the underlying hardware platform, operating system, middleware, and even programming languages. Functionality must be specified using models of computation[3]—such as state machines, dataflow graphs, and hierarchical control flow—appropriate to the application domains at hand.

Researchers must use model-driven code generation to integrate these functional components into the deployment infrastructure. Design-time decisions about protocols and operating points to be used will result in inexpensive but relatively static configurations. Flexible runtime infrastructures can react to online changes and failures, but they must be supported by frameworks and architectures that can operate under the stringent resource constraints of physical computing systems.

Current theoretical foundations for real-time embedded systems are largely built on deterministic and worst-case parameters. Physical computing systems will experience stochastic workloads, and worst-case characterization will force developers to render service guarantees only at very low, average-case utilization levels.[4] Analytical techniques and principles yielding nonzero but practically negligible probabilities of timing and service failures will offer significantly efficient utilization of system resources. Statistical resource manage-

ment theory must be developed to work with the timing and reliability constraints of physical computing systems.

Programming abstractions

Raising the level of abstraction for programmers will be key to the growth of wireless sensor networks. Currently, programmers deal with too many low-level details regarding sensing and node-to-node communication. For example, programmers typically deal with sensing individual pieces of data, fusing that data, and moving it outside the network. Raising the abstraction level to consider aggregate behavior, application functionality, and direct support for scaling issues will increase productivity. Current research in programming abstractions for wireless sensor networks can be categorized into seven areas: environmental, middleware APIs, database-centric, event-based, virtual machines, scripts, and component-based.

For example, consider an environmental abstraction called EnviroTrack.[5] Here the programmer deals with entities found in an application. If the application tracks people and vehicles, the programmer can define people and vehicle entities and use library routines that support low-level sensing functions that detect and classify these object types. These routines can also easily specify the application-level processing associated with each entity type. This lets programmers deal with application-level functionality rather than low-level details.

Given that wireless sensor networks deal primarily with collecting, analyzing, and acting on data, many developers favor a database view of such systems. In this view, a programmer deals with queries written in an SQL-like format. However, real-world data issues—such as probabilistic availability of data, various levels of confidence in data, and missing or late data—can make the SQL paradigm insufficient. Thus, no one programming abstraction for wireless sensor networks will be likely to predominate. Rather, several solutions will emerge, each better suited to certain domains. Results in this area will play a critical role in expanding development and deployment of these networks by general programmers.

Embedded systems software

Software for RTESs poses some of the hardest challenges for developers because these systems must interact with physical devices. Although much work has been done to support model-based development of software and middleware for large distributed systems, the existing technology for RTES design does not effectively support the development of reliable and robust embedded systems.

Researchers must develop a better understanding of how to use software and physical models to design, analyze, validate, and implement large-scale RTESs. Challenging issues for supporting such design paradigms include the following:

- *Model-based implementation and validation.* Developers must explore ways to use the various specification models throughout all development phases so that investments in modeling and analysis pay off directly in the final product development. Potentially promising areas include using models for automatic test and code generation as well as using the logical properties proved at design time for runtime verification.[6]
- *Sharing of modeling artifacts.* In theory, it is easier to share design models than code because models function at a higher abstraction level than code, thus, they are less connected to their target platforms. Little support or effort has been devoted to open model development so far. Correcting this situation should help move software development based on design models into the mainstream.
- *Middleware.* RTESs are increasingly being networked to form complex systems that need various quality-of-service requirements from applications such as automotive controllers, medical devices, and consumer electronics. Key research challenges for middleware include satisfying real-world physical constraints while providing multidimensional quality of service; elevating the abstraction levels at which middleware is developed and validated; developing metrics and validation techniques for evaluation; and supporting the composition of various quality-of-service properties.

Targeted advances in hardware will contribute significantly to making physical computing systems cost-effective enough to be deployed ubiquitously.

Hardware

Targeted advances in hardware will contribute significantly to making physical computing systems cost-effective enough to be deployed ubiquitously. Challenges to achieving this include addressing the disparity between processor and main memory speeds. This has historically led to the use of multiple caching levels and complex pipeline management techniques such as branch prediction. Although these techniques improve the average-case

Conceptual and architectural innovations in hardware will benefit physical computing systems significantly more than will raw performance improvements.

performance of many applications, they also have adverse effects on the worst-case performance bounds that real-time applications can experience. Researchers must design caching architectures that improve both average-case throughput and worst-case bounds to incorporate into future processors.

Similarly, hyperthreading techniques in modern processor architectures must be extended to satisfy the mission-critical and predictability requirements of physical computing systems. Both caching and hyperthreading in RTESs must ensure that resources available to real-time threads are both predictable and controllable so that a thread can be selectively given access to more or fewer resources. Hardware support for resource partitioning, enforced separation of code from data, type checking, and tunable encryption and decryption schemes will significantly improve efforts to ensure resistance to viruses and denial-of-service attacks and to enforce privacy rights.

Active power management is critical to maximizing the life of batteries used in sensor nodes. In particular, we recommend multiple OS, middleware, and application-tunable power settings for processors, communication interfaces, and I/O devices. Nodes must also be able to awaken on reception of authorized commands. Energy harvesting from sources such as vibration, sunlight, and motion will lead to the deployment of new applications in remote and relatively inaccessible areas. Integrated sensors and miniaturized actuators will also reduce size and energy needs while bringing down costs. Standardized interfaces that allow plugging in multiple devices must be developed as well.

Increasingly, researchers are developing novel sensor technologies. Taking advantage of these new devices presents a major research opportunity. New research must address the need for self-calibration and relative calibration of many sensors in situ, rather than in a lab or at a manufacturing site. Meanwhile, ultrawideband capabilities are advancing rapidly. Utilizing these capabilities in physical systems has spawned another active research area.

In short, conceptual and architectural innovations in hardware will benefit physical computing systems significantly more than will raw performance improvements.

System integration

System integration has traditionally been a critical issue in military applications where mission success depends on the trustworthiness of a *system of systems*. With the proliferation of sensor and actuator networks in civilian applications, system integration will be even more critical because of the anticipated scaling required to adapt these systems for civilian use.

System-level composition. Consider, for example, the possibility of integrating the millions of GPS and mobile-communication-equipped embedded systems in automobiles with the metropolitan real-time traffic-management systems that perform traffic-control and emergency-response functions. These diverse systems cannot possibly be engineered in advance with the full knowledge of all the systems they might have to interface with. If we consider the Internet to be the knowledge integrator for personal computing platforms, something equivalent or more powerful will be needed to integrate the future's sensor-management-actuator system of systems. The technical problems will be many times more difficult because of the scale, and the stringency of the requirements for timely information, security, and fault tolerance.

Integrating these diverse systems will require system-level composition techniques. Traditional computer science research has focused on procedure and method invocation as the principal interface between software entities. In physical computing, systems must coordinate with one another at much higher abstraction levels. For example, two systems might need to coordinate their state transitions under uncertainty regarding environmental response and timeliness constraints. In such instances, the communication between systems resembles negotiations accompanying a statement of objectives rather than immutable requests for action. For engineering at the level of a system of systems, developers must implement new techniques for goal and behavior specification, analysis, synthesis, monitoring, and debugging.

Codesign challenges. The prevalence of hardware and software codesign in the new physical computing systems will require more powerful design tools for comodeling, coanalysis, cosimulation, coverification, and co-debugging. Current work on hybrid systems has taken a first step in this direction, but developers have much further to go before they can perform codesign routinely.

For example, special hardware systems implemented in field-programmable gate arrays might need to work with software systems running on stock commercial processors. Given the divergence

in speed between special hardware and stock processors, new approaches must be developed to simulate both efficiently, while enabling the simulation to attain sufficient fidelity in both timing and logical correctness. This is especially important in system-on-chip architectures for which developers need more advanced design tools to explore hardware and software tradeoffs and accommodate interactions with the environment.

Given that physical computing systems must integrate different applications that interface with the external world and are subject to externally imposed timeliness requirements, there will be substantial interference among the applications when they share resources. For example, the scheduling of tasks in one application could affect the timeliness of the tasks in another. Worse, the applications might not be aware of each other's timeliness requirements, thus making resource-access coordination problematic.

Research in open system architecture has shed some light on this problem. Adopting an open system architecture facilitates programming individual applications as if each runs on a dedicated processor using dedicated resources. This well-known concept virtualizes a resource as an abstraction for programming.

However, resource virtualization has traditionally attempted to hide parafunctional details such as timing and fault tolerance from the application to achieve platform independence. With physical computing, this approach loses its viability because applications can now interfere with one another through resource-usage timing and interaction with the physical environment.

Timeliness concerns. Regarding timeliness concerns, a large solution space must still be explored. The crucial issues involve how much task-specific information individual applications must make public for scheduling and when and how often this information must be communicated between the operating system and the applications to make resource-scheduling decisions.

Although there have been advances in this area,[7-9] much work must be done to develop a better understanding of how to balance application isolation and coordination needs, craft open architecture support for middleware, and implement and adopt open architectures that will advance incrementally toward standardization. This is especially important for heterogeneous systems that combine everything from low-end devices—such as Berkeley motes, PDAs, and laptops—with back-end databases and Internet interfaces.

Security and privacy

Wireless sensor networks have limited energy, computation, and communication capabilities. In contrast to traditional networks, sensor nodes often deploy in accessible areas, which exposes them to physical attacks. Sensor networks interact closely with their physical environment and with people, posing additional security problems. Hence, current security mechanisms are inadequate for these networks.

The severe constraints and demanding environments of wireless sensor networks make security for these systems especially challenging.

The new constraints under which they must operate pose research challenges for areas such as key establishment, secrecy and authentication, privacy, robustness to denial-of-service attacks, secure routing, and node capture. Achieving a secure system requires integrating security solutions into every component, because components designed without security become a tempting point of attack.

Consequently, security and privacy pervade every aspect of system design. Consider one of the most difficult attacks to defend against: Adversaries can severely limit a wireless sensor network's value by launching denial-of-service attacks.[10] In the simplest form of DoS attack, an adversary attempts to disrupt operations by broadcasting a high-energy signal. Given a strong enough transmission, the attack could jam an entire system. More sophisticated attacks are also possible: The adversary could inhibit communication through violation of the MAC protocol by, for example, transmitting while a neighbor is also transmitting or by continuously requesting channel access with a request-to-send command.

Security analysts need new techniques for dealing with these simple yet potentially devastating attacks. Many other security-related problems need further research,[11] such as how to secure wireless communication links against eavesdropping and tampering.

Overall, security presents a difficult challenge for *any* system. The severe constraints and demanding environments of wireless sensor networks make security for these systems especially challenging.

Validation and certification

To improve and ensure the quality of RTESs, it must be possible to validate and certify them based on sound scientific foundations. The certification can be done in two steps: first certify that a design has the right properties, then certify that an implementation conforms to the design. With proper scientific foundations, it should be possible to measure

quantitatively how well a system meets its requirements. To achieve this, we need solutions to the following issues:

- *Eliciting formal models from informal requirements.* The development of most systems starts with informal requirements that specify how the system's hardware and software, and the user and environment, are expected to behave and interact. Researchers need to facilitate the elicitation of a design from such informal requirements.
- *Model validation.* The design must be verified and validated to ensure it is correct and consistent with its intended purposes. Much work remains on how to validate that models captured in design artifacts are indeed the ones developers intended.
- *Implementation validation.* The ultimate goal is to check how well an implementation works with respect to the requirements. Researchers must develop metrics for evaluating the degree of validation and the notion of incremental validation. This task is significantly complicated by the noisy and dynamic environments in which the systems operate.

Advances in these areas will help improve reliability and user confidence in low-end RTESs, such as medical devices and consumer electronics. It will also improve the certification process of high-end RTESs, such as avionics and automotive software. This improved quality will result in insurable RTESs, which requires quantifiable reliability, liability, and risk.

We anticipate that physical computing will produce revolutionary changes in the design, deployment, and control of transportation systems, manufacturing, infrastructure protection, power grids, and process control. The beneficial effects of physical computing will be visible across multiple application domains. Aging societies across the world will reap enormous economic benefits by creating technologies for assisted living, while senior citizens and their relatives will enjoy the improved quality of life and greater peace of mind that will result from using these technologies. Physical computing will lead to significant enhancements in the capabilities of emergency responders to save lives and reduce property damage and will better protect infrastructures from natural and man-made disasters. ■

Acknowledgments

We thank those who contributed to the white paper on which we based this article by participating in the preparatory workshop held in Cancun during the IEEE Real-Time Systems Symposium 2003 or by sending us their position statements. Although this article does not reflect all their positions, these contributors include Tarek Abdelzaher, Sanjoy Baruah, Alan Burns, Kevin Jeffay, Walt Heimerdinger, Tom Henzinger, Gabor Karsai, C.M. Krishna, Tei-Wei Kuo, Ed Lee, Jane Liu, Daniel Mosse, John Regehr, Lui Sha, Janos Sztipanovits, Wayne Wolf, Hui Zhang, and Wei Zhao.

References

1. R. Alur et al., "Hierarchical Modeling and Analysis of Embedded Systems," *Proc. IEEE*, Jan. 2003, pp. 11-28.
2. D. de Niz and R. Rajkumar, "Time Weaver: A Software Through-Models Framework for Embedded Real-Time Systems," *Proc. 2003 Conf. Languages, Compilers, and Tools for Embedded Systems* (LCTES 2003), IEEE Press, pp. 133-143.
3. E. Lee, "Embedded Software," *Advances in Computers*, M. Zelkowitz, ed., vol. 56, Academic Press, 2002.
4. H. Zhu et al., "Design Tradeoffs in Networks with Soft End-to-End Timing Constraints," *Proc. IEEE Symp. Real-Time and Embedded Technologies and Applications*, IEEE Press, 2004, pp. 413-420.
5. T. Abdelzaher et al., "EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks," *Proc. 24th Int'l Conf. Distributed Computing Systems* (ICDCS 2004), IEEE Press, 2004, pp. 582-589.
6. M. Kim et al., "Java-MaC: A Rigorous Runtime Assurance Tool for Java Programs," *Formal Methods in Systems Design*, Mar. 2004, pp. 129-155.
7. Z. Deng and J. Liu, "Scheduling Real-Time Applications in an Open Environment," *Proc. 1997 IEEE Real-Time Systems Symp.*, IEEE Press, 1997, pp. 308-319.
8. A. Mok and X. Feng, "Real-Time Virtual Resource: A Timely Abstraction for Embedded Systems," LNCS 2491, Springer, 2002, pp. 182-196.
9. I. Shin and I. Lee, "Periodic Resource Model for Compositional Real-Time Guarantees," *Proc. IEEE Symp. Real-Time Systems*, IEEE Press, 2003, pp. 2-13.
10. A. Wood and J. Stankovic, "Denial of Service in Sensor Networks," *Computer*, Oct. 2002, pp. 54-62.
11. A. Perrig, J. Stankovic, and D. Wagner, "Security in Wireless Sensor Networks," *Comm. ACM*, June 2004, pp. 53-57.

John A. Stankovic *is the BP-America Professor in the Department of Computer Science at the University of Virginia. His research interests include distributed computing, real-time systems, operating systems, and ad hoc sensor networks. Stankovic received a PhD in computer science from Brown University. Contact him at stankovic@cs.virginia.edu.*

Insup Lee *is the Cecelia Fitler Moore Professor of the Department of Computer and Information Science at the University of Pennsylvania. His research interests include embedded systems, real-time systems, formal methods and tools, medical device systems, and software engineering. Lee received a PhD in computer science from the University of Wisconsin, Madison. Contact him at lee@cis.upenn.edu.*

Aloysius Mok *is the Quincy Lee Centennial Professor of Computer Science at the University of Texas at Austin. His research interests include real-time and embedded systems, fault-tolerant and secure system design, and network-centric computing. Mok received a PhD in computer science from the Massachusetts Institute of Technology. Contact him at mok@cs.utexas.edu.*

Raj Rajkumar *is a professor of electrical and computer engineering at Carnegie Mellon University. His research interests include embedded real-time systems, operating systems, wireless networking, and automotive systems. Rajkumar received a PhD in computer engineering from Carnegie Mellon University. Contact him at raj@ece.cmu.edu.*

IEEE

Innovative Technology for Computer Professionals

Computer

Welcomes Your Contribution

Computer **magazine looks ahead to future technologies**

IEEE COMPUTER SOCIETY

- ***Computer***, the flagship publication of the IEEE Computer Society, publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.
- Articles selected for publication in ***Computer*** are edited to enhance readability for the nearly 100,000 computing professionals who receive this monthly magazine.
- Readers depend on ***Computer*** to provide current, unbiased, thoroughly researched information on the newest directions in computing technology.

To submit a manuscript for peer review, see *Computer*'s author guidelines:

www.computer.org/computer/author.htm