PROACTIVE COMPUTING

Human-in-the-loop computing has its limits. What must we do differently to prepare for the networking of thousands of embedded processors per person? And how do we move from human-centered to human-supervised computing?

For the past 40 years, most of the IT research community has focused on interactive computing, J.C.R. Licklider's powerful and human-centered vision of human-computer symbiosis [3]. In tandem with this research has come the creation of an IT industry that is hurtling toward the human/machine/network breakpoint—the point at which the number of networked interactive computers will surpass the number of people on the planet.

We still have a long way to go before Licklider's vision is attained—and are many years from extending per-capita penetration to most parts of the world. However, "missing science" may no longer be the factor limiting progress toward these long-cherished goals. It is reasonable, though perhaps heretical, to suggest that refinements of the existing science base will be sufficient to drive these efforts forward.

It is time for a change. The computer science research community now enjoys a rare and exciting opportunity to redefine its agenda and establish the new goals that will propel society beyond interactive computing and the human/machine breakpoint. In lifting our sights toward a world in which networked computers outnumber human beings by a hundred or thousand to one, we should consider what these "excess" computers will be doing and craft a research agenda that can lead to increased human productivity and quality of life.

David Tennenhouse

Figure 1. The four quadrants of ubiquitous computing.

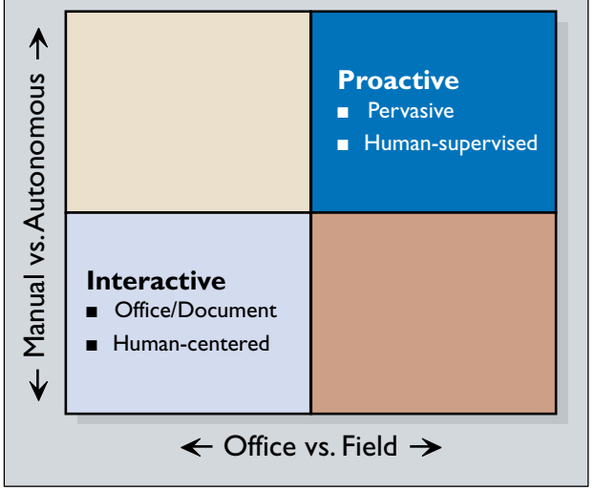
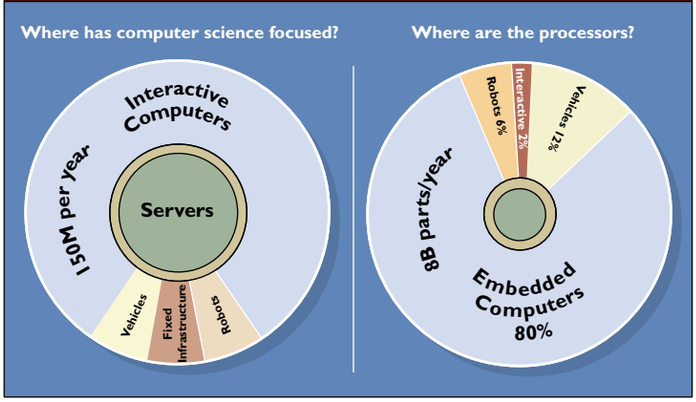


Figure 2. Where will the computers be?



What Should We Do Differently?

Although I lack Licklider's clarity as to what the next 40 years of computation might bring, I am convinced that the first steps toward a new agenda must include: a fundamental reexamination of the boundary between the physical and virtual worlds; changes in the time constants at which computation is applied; and movement from human-centered to human-supervised (or even unsupervised) computing. While some work has been done in each of these areas, focusing significantly greater attention on them will enable a new mode of operation, which I refer to as *proactive* computing.

In this article, I describe three loci for new research activities:

Getting physical. Proactive systems will be intimately connected to the world around them, using sensors and actuators to both monitor and shape their physical surroundings. Research into "getting physical" explores the pervasive coupling of networked systems to their environments.

Getting real. Proactive computers will routinely respond to external stimuli at faster-than-human speeds. Research in this area must bridge the gap between control theory and computer science.

Getting out. Interactive computing deliberately places human beings in the loop. However, shrinking time constants and sheer numbers demand research into proactive modes of operation in which humans are *above* the loop.

There are two simple reasons why we should divert some of our intellectual resources to proactive computing: The vast majority of new computers will be proactive, and these nodes will be the principal sources and sinks of information.

The bulk of the IT industry is presently focused on office automation, e-commerce, and their associated networking. Judging by our current research profile, an independent observer might believe that the distribution of new computers is dominated by the 150 million or so new laptop, desktop, and server nodes that will power the growth of interactive computation.

Although the creation of an industry that consumes such a large number of computers per year is a tremendous achievement, these numbers pale by comparison to the eight-billion-or-more computational nodes that will be deployed worldwide this year. As shown in Figures 1 and 2, the vast majority of these devices will be embedded in other objects. Rather than being in direct contact with human beings, they will be in direct contact with their environments—able to monitor and shape the physical phenomena that surround them.

Since the rate of growth of these embedded devices exceeds that of their interactive cousins, the computer science research community has no choice but to follow the processors and invest a larger fraction of its intellectual capital in this space. In doing so, we would also be following the data, moving from environments in which our sources of information are largely human-mediated to those in which computers directly tap tremendous sources of grounded information concerning the world around them. In fact, the essential reason for having more devices than people, and for having them be distributed, rather than placed in glass rooms, is their intimate connectivity with the physical world—and the incremental sources and sinks of information they provide.

Why Now?

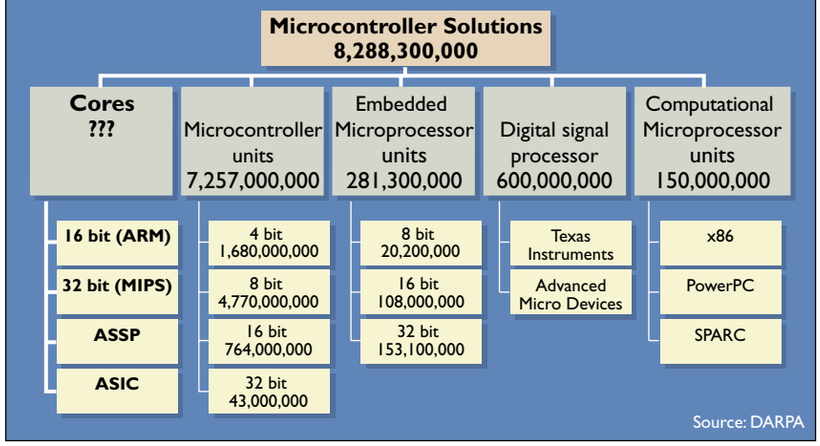
To date, Internet deployment has focused on breadth, expanding the geographic reach of the network to include nodes "in every office and every home." Although this broadening of the Internet will continue at an impressive rate, the number of nodes can be increased an additional 50-fold by reaching down into all of the embedded devices at each geographic location.

This shift toward deeply networked systems represents a profound inflection point that demands our attention because of its sheer scale and systems implications. Historically, the lack of network connectivity has stranded the data obtained by embedded processors and led to rigid software regimes constrained by one-time programmability. However, various efforts are beginning to unlock the information derived by huge numbers of sensors and provide remote access to their actuators (see Pottie's and Kaiser's "Wireless Integrated Network Sensors" in this issue).

Isn't this the same as *ubiquitous* computing? In his 1991 article [8], Mark Weiser, who was chief technologist of Xerox Palo Alto Research Center, forecast that computation would become so ubiquitous that individuals would no longer be conscious of its every application, drawing instead on it as frequently and reflexively as when they reach for a light switch. As shown in Figure 3, the ubiquitous computing space can be divided along two dimensions—one having to do with the degree to which the applications are office-centered, and the other having to do with the degree to which the focus of the computational node and its interfaces is on interacting with human beings vs. interacting with the rest of its environment.

Although Weiser's vision was quite sweeping, few researchers have broken with our traditional emphasis on human interaction. Instead, most research has been confined to the figure's lower-left quadrant. Proactive computing represents an agenda for the exploration of the upper-right quadrant of ubiquitous computing—and the expansion of our intellectual horizons beyond the interactive domain. In short, it's time for researchers to declare victory on office automation.

Figure 3. Projected CPU shipments in 2000.



Let's Get Physical

Herbert Simon, Nobel laureate and a professor at Carnegie Mellon University, identified the importance of bridging the physical and virtual domains quite some time ago [6]. However, the problem has largely been ignored by the mainstream artificial intelligence and computer science communities.[1] Although we have done a marvelous job creating virtual environments and exploring a limited range of interface technologies, such as speech and graphics, the degree to which interface research has been human-centered may have blinded us to many opportunities beyond interactive computing.

The challenge is to develop mechanisms that allow the software properties we have exploited in the development of our virtual worlds to be pushed as close as possible to the physical environment—right up to the converters and transducers that touch the physical world. One example of an effort in this direction is the SpectrumWare project [7] at MIT in which samples corresponding to wide bands of the radio-frequency spectrum are dropped, en masse, into memory, so that traditional radio functions, such as demodulation, can be performed by user-mode application software. Making large swaths of the spectrum accessible to the programmer opens the door to new algorithmic approaches and waveforms that might not have been considered in traditional digital signal processing (DSP) environments.

The following paragraphs address some of the node-level challenges and opportunities associated with getting physical:

Sensors and actuators. One of the distinguishing aspects of proactive nodes is that they will be equipped with sensors and/or actuators. Recent work on microelectromechanical systems (MEMS)

[1] Notable exceptions are our colleagues working on robotics. Although significant work on embedded and real-time systems has also been performed within various engineering disciplines, the approaches taken have been rather rigid compared with mainstream computing.

has led to a great deal of innovation in this space, and it is reasonable to assume that a wide range of small and inexpensive sensors will become available in the coming decade. Although progress on actuators has been slower, MEMS may lead to the first breakthroughs in actuator technologies since hydraulics and the squirrel-cage motor.

Inexpensive network connectivity. Another challenge is the integration of network connectivity that is inexpensive by present standards and tailored to operate in environments in which the vast majority of the network traffic is directly related to the nodes' sample-processing functions. In particular, upper-layer protocol stack and operating system interfaces should be coordinated with the memory subsystem so as to support the efficient exchange of "packetized" sampled data. It will also be necessary to consider the total cost of network connectivity, including the per-node costs of the shared infrastructure, such as base stations. Radical innovation will be required to bring networking costs into line with the $1-per-device price structure of the embedded computing market.

Sample-friendly microarchitectures. Advances in computer science frequently come from finding new ways to trade excess computational capacity for new functionality. Yet practitioners of embedded computing are expending their creative energies in trying to jam functionality into computational nodes that are extremely primitive by the standards of modern interactive computers.[2] Although DSPs have significantly greater processing capacity than microcontrollers, they present a synchronous, rigid programming model that is impoverished and far removed from that enjoyed by mainstream programmers.

The alternative is for processor and compiler designers to create the surfeit of resources that will inspire creative system designs. Proactive computing will leverage new architectures that streamline the processing of the information acquired at individual nodes. For example, one can imagine a uniform approach in which diverse forms of sampled information are automatically packetized, time-stamped, and buffered upon acquisition, allowing their subsequent processing to proceed asynchronously with respect to the external processes being monitored. Similarly, time-stamped sample streams destined for actuators could be automatically clocked out to the digital-to-analog converters, thereby reconstituting signal timing at the edge of the system without programmer intervention. For example, the Berkeley Intelligent RAM effort to develop processor-in-memory technology could lay the groundwork for sample processing environments in which incoming samples are written directly to the memory array for high-bandwidth consumption by bursty software processes [5]. In the reverse direction, time-sliced processes may generate outgoing sample bursts on a faster-than-real-time basis, knowing they will be presented to the physical world at the appropriate rate.

> **How are things different when the interface is being used to supervise thousands of computers and/or millions of knowbots?**

Operating system implications. Sample-friendly processor architectures will enable the bulk, if not all, of the application software to be temporally decoupled from the physical world, allowing the programmer to enjoy many of the advantages of mainstream software environments. For example, software processes operating in faster-than-real-time bursts can leverage traditional mechanisms, such as multitasking, and statistical approaches, whose instantaneous processing requirements are data-dependent. This decoupling implies the need for operating system mechanisms that "virtualize" the incoming and outgoing sample streams and make them available to user-mode application software.[3] The addition of sample-processing interfaces may represent the first significant opportunity for innovation in OS functionality (vs. performance optimization) since the addition of graphics and networking support in the early 1980s.

Aggregating Nodes into Systems

Leveraging large numbers of computers, many of which are embedded in infrastructure, such as roads,

[2]Butler Lampson, a Turing award winner, has suggested that because a design team may have a limited creativity reservoir, it ought to leverage brute force when it is possible to do so, husbanding the creativity budget to finesse problems that would outstrip the growth of computational resources.

[3]Protection mechanisms used to isolate the streams of different processes might be based on concepts like those explored in the Safe I/O work of Ian Pratt, a lecturer at the University of Cambridge Computer Laboratory, and in the user-mode protocol stacks developed within the network research community.

buildings, and transportation systems, will demand new systems and networking technologies that aggregate and share their capabilities.

The Sensor Information Technology Program at the Defense Advanced Research Projects Agency (DARPA) provides a domain-specific example of some of the new directions that are ripe for investigation. Traditional sensor networks are dedicated to a single application and organized hierarchically. However, proactive sensor networks might be based on very different concepts, including:

Sensor multiplexing. Instead of dedicating nodes to specific users and applications, the sensors might be viewed as offering network-based services that can be browsed by authorized users. Since the bandwidth to and from individual sensors will be limited, scaling to large numbers of potential users is likely to depend on intelligent multicasting technology, such as that being investigated by Deborah Estrin and others at the University of Southern California (netweb.usc.edu/SCADDS/).

Inverse and peer tasking. Present-day sensors operate in a master-slave environment in which they are "tasked" by a superior authority. However, future designs might leverage mobile code to invert the traditional hierarchy. For example, a sensor observing an event of interest might launch an applet into the network that changes the tasking of its peer nodes—or even the tasking of server nodes that would have been its superiors in a traditional hierarchy.

Querying and fusion of real-time observations. Since individual users will be able to leverage large numbers of sensors of many different types, and since the capabilities, configuration, and location of these sensors will be dynamic, relational querying may be a viable alternative to traditional sensor tasking. For example, a user could pose an SQL-like query to his or her environment, have the query automatically decomposed into subqueries that are then routed to appropriate nodes within the network, and have the responses fused on their return. In practice, the decomposition and fusion operations are likely to be network-based, leveraging active networks [9] and associated technologies, such as intentional naming [1], geographic addressing, and in-casting.

Sample provenance. For security, legal, and scientific purposes, it will be essential for proactive systems to develop and retain chains of evidence that can be used to convincingly trace sampled information, working backward through any fusion and processing mechanisms to the initial point of acquisition of the information.

Although these sensor network activities are quite novel, they are only the tip of the iceberg with respect to the new systems organizations that will be required.

Let's Get Real

In addition to passing the one-computer-per-person breakpoint, many proactive environments will pass a significant real-time breakpoint, operating at faster-than-human speeds. This is a major change from interactive computing, in which we lock our systems into operating at exactly the same frequency as we do.

Building systems that operate at higher frequencies is not in and of itself a novelty and, in fact, such systems as automobile antilock braking have been fielded in very large numbers with surprisingly good results. However, the growing rate at which faster-than-human systems will be deployed suggests an urgent need to lay the intellectual groundwork for their principled design and analysis.

Software-enabled control. When we speak of a proactive system being faster than human, all we are really saying is that the latency between the system's inputs and outputs is shorter than humans could sustain. However, the real step up in complexity arises because proactive computers will frequently be closing a feedback loop, that is, their actuators will be influencing the environment being sensed. Although a large body of knowledge concerning control systems has been developed over the years, it has been conceived primarily from the perspective of mechanical and electrical engineers and is rather conservative by the standards of software designers. (This perspective is far from surprising; few computer science students are taught anything at all concerning information and control theory.)

Given hundreds of millions of instructions per control interval and sufficient memory to support state spaces numbering in the billions, it should be possible to develop software-friendly approaches that afford increased system flexibility and performance. One promising line of research is exploring a dynamic approach in which context-dependent control systems are created on a just-in-time basis. A background process will engage in the synthesis of potential replacements for the online control state machine, and provision is made for seamless transfers of control from one state machine to another. Other lines of investigation include: faster-than-real-time simulations that race ahead of the system being

controlled to predict its near-term performance under a range of possible inputs; and systems whose behavior is less precisely controlled than at present. In the latter case, the control system ensures operation only within gross bounds that are known to be safe, leaving the detailed operation of the system to heuristic and/or statistical algorithms whose stability is only coarsely bounded.

Network-enabled control. It is likely that the control elements of proactive systems will be interconnected using packet-based networking, thus suggesting another aspect of control theory that should be revisited. Typically, the delay through packet networks is variable, whereas the delay through traditional control loops is fixed. For systems that have a sufficiently large number of components, it may also be important to develop control regimens that tolerate statistical variations in component availability and connectivity, leading to new ways of thinking about fault tolerance in distributed control systems.

Online measurement and tuning. Although low-latency components are not a sufficient condition for faster-than-human computing, they are a necessary one. Unfortunately, much of our hardware and software exhibits far more latency when it is deployed than it does in the laboratory, apparently suffering from "latency rot" after it is fielded. In many cases, this degradation is due to the off-line tuning of parameter values that account for differences in hardware, software, and network configurations that can vary greatly over the life of a system. If we are to get serious about reliable system operation at high frequencies, then we must get serious about online measurement and tuning capabilities.

Let's Get Out

Getting humans out of the interactive loop and into supervisory and policy-making roles is a necessity for systems with faster-than-human response times. Similarly, the sheer numbers of networked computers will preclude human cognizance at the level of individual nodes.

Robotics represents one aspect of autonomous operation in which a beachhead has been established. Unfortunately, much of the work to date has focused on "mechatronics," rather than on software, with the result that some telesupervised vehicles require greater human participation (albeit remotely) than their manned counterparts. However, new DARPA robotics research programs are driving toward yet another proactive breakpoint—the point at which unmanned vehicles will outnumber their human supervisors. These programs emphasize development of novel software capabilities and reusable software platforms that will allow researchers to build on each other's accomplishments.

Robotics systems might outnumber humans by only thousands to one over the coming decade. However, the multiplier attainable by software-based "knowbots" is virtually unlimited, and thinking in terms of millions of active agents per user is not unreasonable. Unfortunately, a great deal of agent research has concerned itself with relatively small systems, rather than working on the larger problem: Given a few billion human users, each of whom is able to generate a sizable agent constituency, we should anticipate interaction spaces involving many trillions of agents. Since these agents will interact with each other as they go about our business, we need to invent technologies that sustain human control over agent-based systems, yet allow agents to autonomously negotiate with each other in ways that honor overall systems objectives and constraints.

> One can imagine the underpinning of computer science taking a leap from deterministic to probabilistic models in much the way that physics moved from classical to quantum mechanics.

Getting humans out of the loop suggests the need for work in three additional research spaces: user interfaces, software, and reuse.

User vs. supervisory interfaces. While considerable work remains to be done on the nature of the human-computer interface for interactive computing, we should start thinking about the interface to proactive computing. How should humans interface with systems whose response times are faster than their own? How are things different when the interface is being used to supervise thousands of computers and/or millions of knowbots?

Software creation. In addition to getting humans out of the operational loop, it will be necessary to reduce human involvement in the software-creation

process. Reduced programmer involvement is desirable because of both the sheer variety of nodes to be populated with software and the difficulty of creating software that interacts directly with the physical world. The interaction problem is rooted in the need to propagate physical constraints through all of the components along the feedback loop(s), that is, all of the software modules interposed between the sensors and transducers. The research challenge is to develop software-creation techniques that automate this process, possibly through direct generation from specifications, as in model-based systems. An alternative approach is to leverage compiler technology that automatically "weaves" constraints into code fragments and "specializes" them, using concepts similar to those recently developed in the Aspect-Oriented Programming project at Xerox PARC [2] and the Ensemble effort at Cornell University [4].

A few good abstractions. The simple reuse of robotics and/or embedded software remains elusive. While component software is often touted as a panacea in this regard, something much simpler and more fundamental might go a long way in the right direction—the identification of significant chunks of functionality that are frequently re-created, along with the specification of a few good abstractions that express their behaviors. A useful analogy can be found in transaction processing, where the informal specification of a single abstraction—the atomic transaction—moved the field from a swamp of unreliable and customized solutions to an environment in which: the solution space is understood; reusable software can be purchased; the programmer is largely relieved of the details of fault tolerance; and, most important, systems realizing 24/7 operation are widely deployed. The development of just one or two powerful abstractions for embedded systems functionality would go a long way toward increased software productivity.

Let's Reinvent Computer Science

Declaring at least a partial victory on the interactive front—especially on office automation—represents a tremendous opportunity to reexamine assumptions and revisit questions that cross-cut many aspects of computer science, including systems architecture, networking, theory, and human-computer interaction. Cross-cutting opportunities ripe for investigation include:

Active technologies. The automated linking of an applet into a running browser may seem like a small thing to the end user and, to a computer scientist, may be viewed as just another demonstration of the power of interpretation. However, the scale and speed with which programs in the field can now be dynamically augmented represents a major change in our ability to assemble, distribute, and update software.

Active technologies—the techniques supporting code mobility—are important in proactive environments for two reasons: The software running on networked embedded processors can be changed on the fly, enabling wholesale transition in thinking concerning embedded functionality; and by launching applets into the network, the lowliest of embedded processors can command the resources of larger systems, providing a very different look and feel to the programmer.

Leveraging large numbers. Information technology is finally reaching a scale where probabilistic methods should play a larger role in systems design. Some of the dimensions becoming significant are:

- Numbers of users and nodes;
- Amount of current information (such as the number of samples) on which each instance of a computation is based;
- Corpus of historical information (such as the number of historical samples) available to build a statistical model of past experience; and
- Scale of the problem and solution spaces being explored.

Over the past few decades a familiar pattern has emerged: A researcher in an application domain (such as speech, vision, or data mining) adopts hidden Markov models to his or her problem; the researcher is pilloried by his or her colleagues; the approach is later shown to be superior and is widely adopted. Given the number of domains in which this pattern has been repeated, it is worth exploring whether some broader conjectures can be supported. In the extreme, one can imagine the underpinning of computer science taking a leap from deterministic to probabilistic models, with hidden Markov models or Bayesian networks supplanting finite state machines, in much the way that physics moved from classical to quantum mechanics.

Various theory and systems researchers are already exploring this space. For example, Eric Horvitz, a Microsoft researcher, has identified a number of key challenges in computer science that might best be attacked using probabilistic methods (www.research.microsoft.com/~horvitz/). One particularly exciting opportunity leverages the feedback properties of proactive systems. Given an algorithm whose statistical performance has been characterized, one might develop an online procedure that recognizes

instances that likely lie in the tail of the run-time distribution, that is, cases where the expected residual runtime of the algorithm is intolerably large. In these situations, feedback mechanisms can be used to kick the system into a different state that is likely to have a shorter runtime.

Reinventing engineering. Although the classical engineering disciplines have embraced computers in many ways, they have not embraced computer science. Instead, programmable devices have been used to emulate traditional solutions. Very little has been done by way of adopting lessons learned in the mainstream computer science systems and theory communities or in rethinking the fundamentals of the various engineering disciplines in light of cheap, plentiful, and networked computation.

Safety, social, and ethical issues. An important class of problems that lies beyond the scope of this article has to do with the broader implications of being proactive. For example, getting physical introduces a range of problems arising from the invasive and pervasive nature of the technologies, while getting real and getting out introduce questions of delegation, safety, and accountability when computers are authorized to make decisions on our behalf.

Curriculum development. The computer science teaching curriculum should be enriched with materials that prepare students to deal with the boundary between the physical and the virtual worlds. As probabilistic methods, information theory, and control theory become essential tools of the trade, they need to be integrated into the academic program.

Conclusion

Over the past 40 years, computer science has addressed only about 2% of the world's computing requirements. Its time to get physical, get real, and get out and build proactive systems. C

The agenda described here was shaped while I was director of DARPA's Information Technology Office, and the article indirectly refers to various efforts sponsored through ITO's research programs (see www.darpa.mil/ito). My thinking in this area has been profoundly influenced by the exceptional DARPA program managers I was privileged to serve with and by the many outstanding members of the computer science research community who gave freely of their time and energy during my tenure at DARPA. The views expressed here are my own and do not necessarily represent those of the Intel Corporation.

References

1. Adjie-Winoto, W., et al. The design and implementation of an intentional naming system. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles* (Charleston, S.C., Dec. 12–15). ACM Press, New York, 1999.
2. Irwin, J., Loingtier, J.-M., Lopes, C., Maeda, C., Mendhekar, A., Lamping, J., and Kiczales, G. Aspect-oriented programming. *Lect. Notes Comp. Sci 1241*, 220–242.
3. Licklider, J.C.R. Man-computer symbiosis. *IRE Transact. Hum. Fact. Engineer. 1,* 1 (Mar. 1960), 4–11.
4. Liu, X., et al. Building reliable, high-performance communication systems from components. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles* (Charleston, S.C., Dec. 12–15). ACM Press, New York, 1999.
5. Patterson, D, Anderson, T, Cardwell, N., Fromm, R., Keeton, K., Kozyrakis, C., Thomas, R., and Yelick, K. A case for intelligent RAM. *IEEE Micro 17,* 2 (Mar.-Apr. 1997), 34–44.
6. Simon, H. *The Sciences of the Artificial.* MIT Press, Cambridge, Mass., 1969.
7. Tennenhouse, D. and Bose, V. The SpectrumWare approach to wireless signal processing. *Wireless Nets. 2,* 1 (Jan. 1996), 1–12.
8. Weiser, M. The computer for the 21st century. *Sci. Am. 265,* 3 (Sept. 1991), 94–104.
9. Wetherall, D. Active network vision and reality: Lessons from a capsule-based system. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles* (Charleston, S.C., Dec. 12–15). ACM Press, New York, 1999.

David Tennenhouse (david.tennenhouse@intel.com) is a vice president and director of research of Intel Corp.