



Trend Wars

Embedded Systems

Dejan Milojicic
Hewlett Packard Laboratories
1501 Page Mill Rd., MS 3U-18
Palo Alto, CA 94304
dejan@hpl.hp.com

This is the seventh and last issue of the Trend Wars department. So far, we have covered hot and controversial trends in operating systems, middleware, mobile agents, Internet, security and privacy, and networking. This last issue is dedicated to embedded systems.

Embedded systems evolved with general-purpose computer systems. They were at the forefront of the use and justification of computer products, the closest to the field. This is where the theories are making or breaking. In the past, most of the embedded research and development was identified with real-time systems and industrial settings, but things have been changing since. With wider deployment of computers, the need for embedded systems has increased. We can recognize this in almost any facet of our lives: embedded systems are in most house appliances, cars, electrical devices, and industrial devices and tools.

This seems to be a long-lasting trend in front of us, of the same or larger impact and disruption than the appearance of the Web. Being widespread poses some hard requirements on embedded systems. They must be as reliable and

robust as other house appliances; as easy to use and as available; connected with other devices, requiring adherence to standards of some kind; and low cost—consequently their development will be defined on a strictly economical basis.

Some of the earlier requirements might not be as relevant in the embedded space. The requirements might have to adjust, making trade-offs, such as size versus flexibility, robustness versus richness of functionality, and power consumption versus performance. The producers of the particular systems will define the exact trade-offs, resulting in a fractured market. Every producer has system software of some kind, typically home-brewed or adapted from one of the many embedded operating systems.

The impact of the many embedded systems produced reflects on the minimization of the software cost. If there are a million embedded systems produced, each one of them worth a few tens of dollars, it is unacceptable that the cost of software is of any significance. The cost in this space is largely dominated by hardware and the savings for hardware are also extremely strict. Because there are potentially

numerous versions and variations of embedded systems, the composability of hardware and software is of extreme importance.

Connecting embedded devices will extend the scalability limits of today's systems even beyond the Internet's global scale. One user can have hundreds and thousands of embedded devices, disrupting traditional networking and, in particular, addressing techniques. Furthermore, it would not be possible or economically viable to connect all these devices by traditional wired technologies; therefore, wireless will become an attractive alternative, opening up new research and development areas.

What are the other long-term trends and impacts of this technology turn? What other implications does it pose to the current computer science and technology? This and other questions are addressed by six renowned researchers in this field. With this last department issue, we thank you for traveling with us through the trend wars in the past year.

—Dejan Milojicic



Brian Lazara

Embedded systems have been around for a while. Why the sudden change in interest? What circumstances were missing in the past that exist now?

There is a huge interest in embedded systems now, and they've been around for

a while. However, I don't think they've been quite as widespread in the past. The change in interest is just the fact that they're everywhere—the Internet is certainly part of it. You can now have smart devices interconnected and networked together, which makes it much more useful to put a micro-processor in various kinds of appliances and other such devices. You wouldn't have thought about doing that before.

Questions

1. Embedded systems have been around for a while. Why the sudden change in interest in them? What circumstances were missing in the past that exist now? Is embedded systems R&D only a passing fad, or will it be with us for a while?
2. What are the biggest challenges for ubiquitous embedded systems in the future? What features will be crucial for achieving acceptable embedded systems: security, real time, scalability, or high availability?
3. What are the most promising embedded systems applications and the dominant deployment areas? Are there any killer applications that will make it or break it?
4. Are standards needed for embedded systems? In which subareas?
5. How do existing de facto standards, such as systems software, interplay in the embedded arena? What about other legacy software, middleware, tools, human interfaces, or databases?
6. Who are (will be) the dominant players in this area? Startups, universities, big companies?
7. What is the direction of the future embedded operating systems: small OS growth (such as VxWorks) toward richer features and support or migration of general purpose OSs, such as Linux and Windows, into smaller sizes and less functionality? Will size and the amount of functionality matter in the future? Will OSs be relevant in future embedded systems, or will it become an invisible solution, glued to the application and the rest of the system software?
8. Are reconfigurability and composability crucial for embedded systems? Will hardware prices be dominant and require customization, or will general-purpose hardware, in particular processors, be more used in embedded systems?
9. How will networking impact embedded systems, in particular, high-speed communication links and wireless?

What are the biggest challenges for ubiquitous embedded systems in the future? What features will be crucial for achieving acceptable embedded systems? Security, real time, scalability, or high availability?

I say all of the above. I think that as embedded systems become more powerful, the kinds of things you can accomplish in an embedded environment will be much more complex than they were 10 years ago. Being able to provide an environment that is secure and highly available while still delivering deterministic real-time characteristics is very important.

What do you think will be the dominant applications for embedded systems in the future?

I think the market is so fragmented that you can't have one dominant killer application. They're not general-purpose platforms that embedded designers are developing. In a lot of cases, they're very single-purpose, and I don't think that you can have a killer app that will cross the entire embedded market. Certainly, there are some things that stand out—the use of TCP/IP, embedded browsing, and Java will all be important—but no, I don't think there will be one killer app that you will be able to point to as dominating embedded applications.

If there isn't a single app, what do you think are the most promising application areas?

Networking, networking, and networking. Whether it be an embedded microprocessor in some kind of server I/O device, network infrastructure equipment, or consumer devices, networking protocols and applications will dominate embedded applications.

Are standards needed for embedded systems, and if so in which subareas?

Standards in networking are of great importance. Without them, we wouldn't be where we are today. There have been attempts to make standard application programming inter-

faces (APIs) for operating system services and in some areas that will be important, but standards will truly be critical at the communications interfaces. In an open third-party environment, you're going to need some API standard so that people can write their applications for this open platform, but the majority of embedded systems are not open. However, those devices do need to communicate over a wire or some kind of median to another computer somewhere, and it's on those interfaces where standards are critical.

What do you think about de facto standards, such as Microsoft or Java products? Do you think that these will find their way into embedded systems or that embedded technology will grow from scratch—from new applications to new standards?

Both. In certain kinds of devices, whether it be a set-top box or a home appliance, some of the de facto standards will be important to embedded applications. In other areas, new standards that are specific to the embedded world will come solely from the embedded landscape.

Who do you think will be the dominant players in this area? Is it startups, small companies, big companies, or universities?

All of the above. That may seem like a cop out, but the embedded market is so widespread and diverse that you're going to see large contributions from all of these areas.

Specifically in operating systems, what do you think will be the major development trends? And will operating systems be important and relevant in the future of embedded systems?

The operating system will definitely be important and relevant, but the companies behind the operating systems and the kind of things they can offer for the embedded developer will be more important. Tools, middleware, support, and consulting will all need to be available from a single company to provide embedded developers with a value chain that will bring their

products to market quickly. Most embedded application developers do not have the time or desire to be in the OS business. That means getting together with a provider you can count on. A company doing a set-top box or some other smart device won't want to be in the OS business either. They're going to look to other companies to provide the services that they need in these complex devices. An embedded device will no longer typically have 10,000 or 15,000 lines of assembly language; now there could be up to three million lines of code all interacting with the environment. That means reliability and portability along with powerful development tools are key to what developers will look for in an OS provider.

Are reconfigurability and composability crucial for embedded systems? Will hardware prices be dominant and require customization, or will general-purpose hardware, in particular processors, be more used in embedded systems?

General-purpose processors will be used more in embedded systems. Building a general-purpose platform provides freedom in a fast-changing marketplace. A printer today might be a file server tomorrow. Designing too much noncustomizable hardware into your embedded system will mean that the system isn't flexible to the changing marketplace.

How will networking impact embedded systems?

Embedded systems have already been impacted by networking in a huge way. Part of the value proposition for having a 32-bit processor in your device is that you can then include a network stack along with various network protocols and have that device be connected to the outside world. This has completely changed the versatility of an embedded device. If you can connect a widget to a network, you can remotely manage that widget including things like automatic upgrades of your application. This is a powerful feature and one of the reasons why the embedded market is exploding the way it is.

Brian Lazara is the director of Wind River's Server Products Group that focuses on embedded solutions for high-performance server I/O applications. He has 14 years experience in embedded software development. He was a primary contributor for Wind River on OS development for the Mars Pathfinder project. Before joining Wind River, he worked for General Electric, building high-speed digital signal processor applications for sonar test equipment. Contact him at brian.lazara@windriver.com.



John Stankovic

Embedded systems have been around for a while—why the sudden interest in them? What circumstances missing in the past exist now?

Many factors have come together simul-

taneously. In particular, wireless is becoming very prevalent; it's now almost pervasive—we have the ability to compute anywhere. Internet appliances are also driving this phenomenon, as are the improvements in sensors, actuators, and handheld devices of various sizes, and global Internet accessibility.

Other hardware advances have occurred; we have low-power devices and high-speed digital signal-processing chips, which have spurred the field forward. I think there's a lot of potential for microelectronic mechanical systems, and that's causing people to think more ambitiously about the future and how these cheaper and smaller sensors and actuators can create ubiquitous smart environments. Commercial embedded systems are also making a difference—we can use wireless devices to send audio and video and interact in ways pure text modes can't provide. On the software end of things, we're seeing micro-browsers, middleware standards, and devices that can communicate with each other through a simple communications protocols. Many of the embedded systems in the past were in defense, manufacturing, or safety-critical systems, but now this technology is found in many commercial ventures.

What are the biggest challenges for ubiquitous embedded systems in the future?

The obvious ones are security, real-time, scalability, and high availability, but I think other key challenges exist, such as what I call performance-based interoperability. Although these complex, ubiquitous systems are glued together with layers of protocols, they still have time constraints and other performance demands that impact how the system will perform, and thereby how it will be accepted by the public. For example, if I have a deadline for sending videos across multiple links, is it really going to get to the people wanting to watch the video in a manner that they can view the video properly? Solving this issue requires meeting time constraints and going through layers of protocols, software mappings, and switches from one kind of network to another, and through layers of software. If the result is a poor quality video, people won't accept the product.

Also, for embedded systems to be universal they must be easier to use, and we're starting to see that. For example, with e-mail-enabled phones, you might want to go through the Internet and download your e-mail; but you still have to punch in a URL using the keypad on the phone and then the result is three tiny lines of text on the screen, which is not too exciting. Yet some people like it, and they're using it. Better interfaces will make this application more prevalent.

Another challenge for smart environments is safety. For example, in a smart university, you won't want to see doors opening and closing at the wrong times, or windows slamming on somebody's hand. Smart environments must be safe environments. In the end, people won't want them if they're not safe, available, and reliable. In fact, smart environments must be as reliable as, say, the power grid. We come in every day, we

turn on the lights, and the electricity is there. We need the same kind of performance from these smart spaces.

What do you think are the most likely applications? Is there any need for a killer application or will this happen anyway?

I don't think there's a need for killer applications, because I think there are a lot of them—I'm kind of an optimist on this issue. Systems like a smart house will push this field forward. I think smart forms of entertainment are going to be key driving applications. It is easy to imagine systems like smart classrooms, universities, hospitals, and airports, and I think that they will come along piecemeal, improving over time.

What about standards? Some of them have proven successful, and some have failed miserably. Do you think embedded systems really need standards to drive them?

I think we definitely need standards. You can see this by looking at the networking layer and how we're going to get all these devices to talk to each other. There's an alphabet soup of telecommunication technologies, and they all must interoperate, using complex telecommunications protocols to get the data from A to B. Without standards and standard interconnects, it's not going to work. Even at the software level, things have to interoperate. I don't think there will be any magic solution, because there are many varieties of standards and embedded systems are pretty broad; everything from your wristwatch up to the air traffic control system. On the other hand, your wristwatch isn't the same as the air traffic control system. So I think quite a few different standards will be necessary.

Who do you think will drive standards development? Startups? Universities? Research centers? Something entirely different?

Because of this field's size, the dominant players are going to be lots of different groups, and I believe the telecommunications companies that provide the infrastructure are crucial. This isn't just WorldCom or Lucent, but the companies that are building handheld devices and even the hardware companies that are building DSP chips. Software companies such as Microsoft, IBM, HP, and other major corporations all have major products in smart spaces, so I believe the large companies and manufacturers will play a driving role.

On the other hand, because the market potential is so great, startups will also play a role. They will generate Internet appliances, handheld devices, and different kinds of smart robots that we can't even imagine today. Universities will play the role they normally do—coming up with new ideas. My guess is that universities will be more attuned to the per-

formance-oriented aspects and new products, rather than standards.

Do you think that future operating systems for embedded systems will be built from scratch, or do you think that some version of Windows will propagate down and be adapted to their needs? Or do you see a third solution?

If you took Linux and broke it down until, say, the only thing left was a dispatch table and two tasks that always execute in a fixed order, is that Linux anymore? When does Linux lose its basic flavor? I think that what's going to happen to operating systems are that they're moving toward a component view, and they're making it so you can select which components you specifically need. But when you get down to smaller devices, the components become much, much smaller—you can take just the dispatcher and the interrupt handler, call that the OS, let two user applications run on it, and this combination of software is your software product. There are commercial ventures

doing component-based OSs, but how good they are remains to be seen. At least they're trying, but although the trend is heading to smaller components, a good analysis method is still missing. Once you collect these components and put them in a device, how is it going to work? Are the components going to actually meet deadlines? Are they going to have some difficult-to-detect dependencies that exist across the components? Personally, I am trying to apply Kiczales' ideas on aspects to embedded

systems. I think that this has excellent potential to address hidden dependencies across components.

What about configurability and composability? Which will dominate the future—general-purpose propagating down or special-purpose spreading up?

I think they'll meet in the middle somewhere for most systems. If I'm building a wristwatch with software in it that I don't expect will change, and I want to sell a million copies, then I'm going to configure it offline and optimize it. But if I want a family of next-generation watches, and I want to quickly change some features in the watch to make a new watch in a product family, then I want to build on what I have, using software component libraries. On the other hand, if we're looking at smart spaces, they have to evolve and continuously upgrade themselves. You need on-the-fly reconfigurability and composability; these systems have to be up and running all the time, which is a complex problem that requires more general-purpose kinds of components. Let me point out that I do not feel that DCOM, JavaBeans, or CORBA components are appropriate for embedded systems. We need simpler components that are built to address both functional and non-functional

There's an alphabet soup of telecommunication technologies, and they all must interoperate.

aspects such as memory, power, dependability, safety, security, and real-time constraints.

What about networking? You emphasized at the beginning the importance of wireless. What about high-speed communication links?

Almost all embedded systems will be networked. There aren't going to be very many things that can just stand alone anymore. You can see this in the home—people want to network their TVs, VCRs, stereos, refrigerators, and so on; networking is here to stay. By the way, I think this is one of the driving motivations for this new generation of embedded systems.

There is a vast difference in bandwidth and requirements in the networking of embedded devices. In a car, you don't need it to be too flexible. You can design it ahead of time—the four brakes communicate over a local area network, which is also well designed, laid out, scheduled, and understood. But in a smart environment, you might have some simple sensor that indicates somebody walked through the door or some complicated device that sends video, which needs tremendous communication bandwidth. I think that for many systems there is almost no distinction anymore between whether an embedded system is stand-alone or on a network.

Any important observations that we can all learn from?

Only that this is a tremendously exciting time that is rapidly moving to the next phase of how we're going to use the Internet. I'm not sure how all of this will affect society or how long it will be before we really have effective smart spaces. It probably won't happen for a while, but many people are working on it.

John Stankovic is the BP America Professor and Chair of the Computer Science Department at the University of Virginia. He also serves as an elected member on the Board of Directors of the Computer Research Association. He is the Editor-in-Chief of both the *IEEE Transactions on Distributed and Parallel Systems* and the *Real-Time Systems Journal*. His research interests include distributed computing, real-time systems, operating systems, and distributed multimedia database systems. He is currently applying his research to developing smart spaces. He received his PhD from Brown University. He is a fellow of both the IEEE and the ACM. Contact him at the Dept. of Computer Science, Univ. of Virginia, 151 Engineer's Way, PO Box 400740, Charlottesville, VA 22904; stankovic@cs.virginia.edu.



Wolfgang Schröder Preikschat

Embedded systems have been around for a while, so why the sudden interest in them? What circumstances missing in the past exist now? Is embedded systems R&D only a passing fad, or will it be with us for a while?

I think it will be with us for a while. However, I'm astonished that the computer science field hasn't regarded embedded systems that much until just recently. Market demand is the driving factor for interest in embedded systems, and it's forcing the rest of the computing industry to follow.

What are the biggest challenges for ubiquitous embedded systems in the future?

Correctness—getting systems software and applications to run correctly, especially because they're used in many safety-critical areas.

Another big issue will be scalability, meaning that the industry must face the challenges of designing complex software that scales well with existing microcontroller solutions for embedded systems.

What do you think will be the most promising applications in embedded systems?

That's difficult to say. There will be applications in the home and entertainment areas, but we'll also see more in the automobile sector. There are some predictions that by 2005 about 70% of a car's cost will be due to its electronics, and most of this will be due to embedded systems technology. Today, only 30% of a car's cost is due to electronics. So we will see a big shift in embedded systems technology in two sectors: applications for the automotive industry and the much more visible embedded systems in personal digital assistants, games, and so forth.

Do you think there will be a need for a killer application?

An embedded system is a computer-based system that is hidden inside a product other than a computer. So the product is the main issue—if the product is there, then the embedded system must be there. We don't see the embedded system, so we don't need to focus on a killer application unrelated to the basic product.

What about interoperability and reusability? Do you think standards are needed in this area, and if so, what is their relevance?

Standards are a big issue, especially because a single company or manufacturer does not build the larger embedded systems. The supplier industry drives this area, so the manufacturer (say, for a big car manufacturer) takes embedded system components and pulls them together to create a big product. For this to work at the technical, hardware, and even software levels, you must use standards just to be able to have all these pieces work together.

Who is driving the field right now and who will do it in the future? Startups? Universities? Big companies?

I think companies will drive growth. We mainly use embedded systems because these products are computer-based, and companies make products. However, the companies don't necessarily have to be large—small and medium-size enterprises, especially the supplier industries, will also impact this field. I don't see universities being dominant here. They rarely create products—they mainly provide research, which is certainly important, but industry controls manufacturing and production.

Focusing only on embedded operating systems, what do you think is the direction of development? Do you think that, similarly to other solutions, we will start from special-purpose small operating systems, or do you think that we could downscale some operating systems?

It depends on the embedded system's complexity. If you look at the microcontroller market, say, the 8-bit microcontrollers, the software that runs in these systems has to cope with only a few kilobytes of memory. A single person might easily write an operating system for such a controller within a few months.

But if you look to the more complex embedded systems that have distributed embedded systems consisting of maybe 50 or hundreds of complex individual nodes, then we certainly have to write more and supply more system software and even application software to make all these systems run. A single person cannot really do this as he or she could for a small system. In this case, we have to think about using operating systems such as Linux, how to reuse certain components of these systems, or how to scale these systems down with respect to the embedded system's demands.

However, it's extremely important to take an application-oriented view. There's a mix between the smaller systems, where we can develop individual solutions off the shelf, and the more complex systems.

In either case, the OS wouldn't be visible; it would just be part of the whole solution.

Right. That's exactly what I said before about embedded systems being hidden inside a product. The operating system is part of the embedded system, so nobody will see it.

Do you think that, to keep the price of development low, we will be able to apply reconfiguration and composability techniques to embedded systems? Reconfigurability and composability haven't been that successful, at least in the operating system arena.

We certainly don't have a reconfiguration aspect right now, for example, if you look at fuel injection control for a motor engine. But in some embedded systems there are reconfigurability issues. Just look at the Pathfinder mission to Mars; it would have never succeeded if we hadn't been able to debug, correct, and reprogram system software. It was crucial to have an operating system running inside the Pathfinder that provided services for reconfiguration.

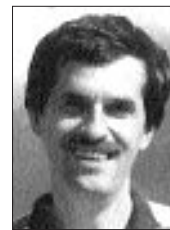
Cost will always be a big issue. If you look at software, the

only cost is engineering, but for hardware, you have to pay for every piece of metal and every part of a circuit. Every time you build or copy a piece of hardware, you have to pay for it. Almost 60% of all the processors built this year are 8-bit technology, not because we love mid-1970s technology, but because it's the price. The only thing that changes that is volume—the more you can build, the cheaper it is. It's an embedded system's cost, which hardware dictates.

What do you think of the impact of connectivity to the future of embedded systems? Will they always be connected?

To give an example, let's look again at the automotive industry. Today, cars consist of about 50 or more microcontrollers, which are interconnected by dedicated networks. Thus, connectivity is a very big issue—there is no longer a situation where microcontrollers only perform individual tasks independently of each other. This is why there's an increased interest in parallel and embedded systems.

Wolfgang Schröder-Preikschat is a professor of computer science (computer networks and operating systems) at the University of Potsdam, Germany and a professor of computer science (operating systems and distributed systems) at the University of Magdeburg, Germany. He has published a number of papers and a book on operating systems design for distributed-memory parallel machines. His main research interests are embedded, distributed, and parallel operating systems; object-oriented software construction; communication systems; and computer architecture. He received a PhD in computer science from the Technical University of Berlin. He is a member of the ACM, FiFF, GI, IEEE, and VDE/ITG. Contact him at Otto-von-Guericke-Universität Magdeburg, Fakultät fuer Informatik, Institut fuer Verteilte Systeme, Universitaetsplatz 2, D-39106 Magdeburg, Germany; wosch@wotan.cs.uni-magdeburg.de.



Liviu Iftode

Embedded systems have been around for a while. Why the sudden change in interest in them? What circumstances were missing that exist now? Is embedded systems R&D only a passing fad, or will it be with us for a while?

The excitement in embedded systems potential is mainly produced by one ingredient that has been recently added to the embedded technology: networking. Networking will bring convenience to embedded computing, which the Web brought to the Internet. Interconnected—especially wirelessly interconnected—the embedded systems become easier and more exciting to use, will require less attention, and can be of more help. They will invade our life in countless incarnations with applications hard to imagine today. Besides networking, other factors, both technological and social, such as low-power processor advances, the Web, and the increasing demand for

personalized services, are also responsible for boosting up the embedded technologies and the interest in them.

Do you think embedded systems are here to stay?

They will stay because embedded systems' evolution is the realization of the long-awaited dream of making the computer transparent and ubiquitous. This is why I believe embedded applications will become the driving force of hardware and software technologies.

What do you think are the biggest challenges? You spoke about opportunities, but what are the challenges for achieving this ubiquitous applicability?

In my opinion, the main challenge is to define a distributed computing model for networked embedded systems. Networking these devices is just the first step. The ultimate goal is to make them cooperate to combine or aggregate their functionalities or resources. Their number and variety is so large that a traditional distributed model simply cannot be applied without causing an overwhelming programming overhead. To keep programming at such a scale manageable, the new computing model must tolerate incomplete results, partial synchronization, and weak consistency. So far, researchers have proposed scalable solutions for simple cooperative tasks such as routing using content-based addressing.

Interoperability is an even bigger challenge. The diversity of embedded devices on which we rely will make our everyday life extremely difficult, if we cannot have these devices silently cooperate by exchanging data and tasks. Finally, fault tolerance and security are traditional challenges for any distributed system, and they will ultimately determine the acceptance of embedded technologies by society.

What are the most promising embedded systems applications and the dominant deployment areas? Home, entertainment, or medical?

Embedded systems have already been in use for a long time, but they have been sporadically networked. First, networked embedded systems that will affect our lives will appear in our homes, very likely in our kitchens, and will be connected to the Internet. The first significant milestone the non-IP networked embedded technology will have to mark is the automotive industry. Everyone uses cars and expects driving to become easier, more pleasant, and exciting. We can achieve this if the car's embedded computer will cooperate with other peers and information authorities on the road. Medical applications are extremely important, but I think that it will take a while before embedded systems will be confidently used in hospitals or in the human body.

You don't think there is a need for a special killer application that will make or break embedded systems? They are just going to happen anywhere, and there are so many application domains that embedded systems will become widespread anyway.

I think that it is a matter of inducing a taste in society for embedded devices to become necessary. Interestingly enough, the first generation of embedded systems were not as visible—few people realized that there was a microprocessor in a cellular phone—because it didn't change their use. The cellular phone remained a single-function device until recently. Once embedded systems generate a difference in the way the user perceives the device, people will expect other devices in their homes, offices, cars, and so forth to be similarly rich in their functionalities.

In the past, these embedded devices were typically related to standards. Do you think this will remain the trend?

In general, standards were both a desire and an impediment in computer technologies. Given the expected variety of embedded systems and the desire for interoperability, it is difficult to imagine an evolution without standards. The fact that commodity electronics will incorporate embedded systems is an indication that industry will demand standards early on.

Given the expected variety of embedded systems and the desire for interoperability, it is difficult to imagine an evolution without standards.

How do existing de facto standards, such as Microsoft products and Java, interplay in the embedded arena? What about other legacy software, middleware, tools, human interfaces, or databases?

The legacy software will "survive" in the back ends, to which the embedded systems will be connected, but it is unlikely that traditional de facto standards will be of any use in the next generation of embedded systems. There is a marketing strategy for selling traditional standards by promising that they will extend to growing embedded systems technologies, but the characteristics of the existing standards are quite different from embedded systems requirements. The IP protocol, for instance, will not be appropriate for certain embedded systems networks because handling millions of volatile nodes using individual addresses doesn't make sense. Instead, an IP-less addressing scheme based on properties or content will be preferred. Similarly, it is not clear that even the Java Micro Edition is small and simple enough to be deployed on a large variety of embedded devices.

Who are or will be the dominant players in this area? Startups, universities, or big companies?

I think that industry already controls this field to a large extent. Moreover, unlike the Web, embedded technologies are likely to flourish more in big companies than in startups because they require substantial investments, infrastructure, and long-term commitment. The demand will be so large that we cannot expect to have only a few players. The driving force in establishing the influential players will be the alliances

between the big customer companies and embedded systems manufactures. It will matter, for instance, what embedded processors General Motors will use in their cars and trucks.

Do you think there will be a new paradigm of system software for embedded systems? Do you think it will start from scratch or from one of the small existing operations that will grow into desired functionality? Or do you think some of the de facto standards such as Linux or Windows C will downgrade to what the market needs?

Following up on what I just said, in the short run, both Windows and Linux will be downgraded to micro versions, but this is largely motivated by marketing rather than technical factors. In fact, we have a real vacuum of software for the networked embedded systems, so this is where university research can play an important role. Actually, the Serendipity project at Rutgers aims to develop a software platform for cooperative computing over embedded systems networks. We plan to develop a simple OS core embedded in an energy-aware virtual machine along with a content-based communication protocol.

What do you think about the configurability, composability, and other general software development techniques in this area? You already pointed out that most of these solutions, such as operating systems and applications, will be merged. Do you think that different configurations of the same software and software techniques for actually building these solutions will be important?

Yes, I believe that embedded systems are an ideal platform for component-based software technologies. Moreover, the next generation of embedded devices will have to be programmable and reconfigurable. This again gives a strong incentive for the design of a flexible, yet simple embedded systems software architecture.

Liviu Iftode is an assistant professor of computer science at Rutgers University. His research interests include distributed and parallel systems, operating systems, networking, and mobile computing. He has a PhD in computer science from Princeton University. Contact him at iftode@cs.rutgers.edu.



Rodger Lea

Embedded systems have been around for a while, why the sudden change in interest in them? What circumstances were missing in the past

that exist now? Is embedded systems R&D only a passing fad, or will it be with us for a while?

I think there are two classes of embedded systems. Deeply embedded systems are the kind of things you'll find in process control and factory automation, for example. They are not really in fashion at the moment, and I think they'll stay that way, because the amount of technology involved is quite simplistic.

Over the last few years, lightly embedded systems have become much sexier, because they're becoming much more technologically advanced in terms of their processing capabilities and their ability to communicate. We're seeing the capabilities of those devices migrate toward more general-purpose, real-time systems and, therefore, more general-purpose computing systems. As they do that, they become more amenable to some of the research that's going on and therefore more interesting.

What are the biggest challenges for these lightweight embedded systems?

Real time, security, and scalability are important issues that people have been addressing for many years and will continue to address as these systems evolve.

For me, the major challenge for embedded systems design and development has to do with tools and process. The tools associated with embedded systems are not very strong; they could be better. There are tools available for just general-purpose programming and debugging. However, as these devices get more sophisticated, start to communicate with other devices, and build up networks of embedded systems, we'll need distributed programming tools.

What are the most promising embedded systems applications and the dominant deployment areas? Are there any killer applications

that will make it or break it?

No, I don't think there is a need for killer applications. Our daily lives are getting more sophisticated because of the technology we have around us, both in the workplace and the home. Those pieces of technology require embedded systems, and therefore there's already a demand. In a sense, the killer application is just the complexity of life, and because of that, there isn't a need to develop new applications that will drive embedded systems use. I think that's happening naturally.

This is certainly true of the embedded systems we use in the audio-visual industry.

Given this mention of interoperability of distributed embedded systems, what is your perspective on the need for standards and their effect in this area?

The next big direction for embedded systems is going to be the connection of different embedded systems. We're seeing this to a certain extent already, but I think we're going to see an increasing move toward the dream that a lot of people have of ubiquitous systems, in which devices embedded around the home

and the office are communicating. Communication, and the associated control protocols, is the area that's crucial for standardization. Without clear standards, it becomes difficult to build ubiquitous devices connected to our computing systems.

Building tools that let you program, monitor, and use these systems is the crux of the embedded systems and embedded systems software. They let you interact with those pieces of technology, and they will become important as part of that communication standard.

A lot of people are probably saying, do we need standard pieces of operating system, or do we need standard applications? I think these devices are specialized; the issue is not how we build the software for those devices, in particular, but how those devices communicate with other devices.

What about de facto standards such as the Java Virtual Machine? Do you think that these general-purpose tools and environments will propagate down to embedded systems, or do you think there is a need to build new tools from scratch?

That's a question a lot of people are asking. The main characteristic of embedded systems is their diversity.

It's difficult to understand how general-purpose and de facto technologies, such as Windows or Java, will solve all the problems in that space, because those pieces of technology rely on an underlying piece of technology that has certain general-purpose, well-known characteristics. If you look at the Wintel platform to date, it's standardized, and there's very little you can change there. The embedded systems domain is much more diverse, and therefore, there's less area for those pieces of technology to play.

That's not to say that they won't have an effect. You've probably seen recently that Windows CE has been adopted by one of the large cable operators and is being used in set-top boxes. Now, set-top boxes are a high-end version of an embedded system, but they do have embedded system characteristics, so it is possible for existing pieces of technology to play.

At the end of the day, what's required is a piece of software technology that is capable of being general purpose and exploiting the underlying hardware—both general and specific. Achieving that type of balancing act requires software that adapts and configures to its environment.

Who do you think will be driving this work? Who are or will be the dominant players in this area?

There are obviously a lot of established players in this space already: the hardware manufacturers and the chip manufacturers—the Texas Instruments and Motorolas of this world. There are also a lot of established software and tools vendors, such as Wind River of the newer Linux players.

These players will continue to evolve, but there is room in particular areas for improvements from new players. Whether they will be universities, other big companies, or startups, I don't know.

Universities have a lot to offer simply because they've been looking at these kinds of systems more from a real-time perspective. These systems are now becoming more open and accessible. Previously, they were very closed systems; manufacturers would basically develop some technology and put it

into a product. That is starting to change, so I think there are opportunities for universities, but everyone needs to be aware that there are a lot of existing players.

Do you think that it will be the companies that develop general-purpose or even special-purpose computers? Will there emerge special companies dedicated only to embedded devices, or will it continue to be a fractured market?

I think it will continue to be a fractured market. If you look at the market today, there are companies like mine that are large players. We have a range of products, and we put technology together. We also produce hardware for our particular devices. Then, there are companies whose role is really just developing the lower-level hardware and software. I can't see that changing very much.

The embedded systems technology that goes into a printer is very different from the embedded systems technology that goes into a process control application, a car manufacturing plant, or a hard disc recorder. They have unique characteristics, and so it's difficult for any one company to capture and corner the market. That diversity will always lead to a segmented marketplace with different solutions for different areas.

At the very beginning of our discussion, you mentioned networking and distributed embedded systems. What is your perspective on the future of the high-speed links, wireless versus wire communication, and the role these will play in the embedded space?

An area we've been working on in terms of home networking for several years now is developing the home audio-video interoperability (HAVI) home networking infrastructure based on IEEE 1394. Sony has an entire lab dedicated to interconnecting architectures. We're seeing embedded devices being connected to other embedded devices with high-speed links in what I call a complete system structure.

Probably more importantly, getting devices to connect in the home, office, or factory is going to change the type of software we develop and the type of capabilities that we offer through embedded systems. For me, that's probably the most interesting and exciting area. Once you connect devices, and you can move toward this notion of ubiquity and ubiquitous devices you can build much more interesting and natural applications from a consumer's viewpoint.

We're interested in trying to provide a natural environment for consumers to interact with their devices. Today, the consumer tends to interact with a device through its front panel. In more sophisticated cases, they go through something like a TV or PC display. We're looking at connecting many devices and providing natural and friendly access to those devices from different parts of the home so that it's not necessary to be interacting physically with the device. For example, you can turn on your CD player from anywhere in your home by using speech input or some other kind of multimodal input. In a sense, you won't care where these things are anymore. High-speed communication links, certainly wireless ones, will become important for embedded systems devices, particularly in the home over the next five years.

Do you think that this opens up many social aspects with humans suddenly surrounded by all kinds of wearable embedded devices?

We've already started to do demographic studies in our lab. We've gone into people's homes and looked at the way they interact with technology. The model of how consumers work with technology is changing, and technology is becoming embedded into nearly everything we interact with. Consumers are not aware of that, just that devices are smarter in some way. Exposing that smartness is something we have to work at.

As we move toward multimodal input models, the user interaction model for consumers becomes much more complicated. The idea of talking to something to make it work is still strange for consumers, and acceptance will take time.

Rodger Lea is Vice President and Director of Sony's Distributed Systems Lab, where he has been helping Sony formulate its digital home network strategy and the role of interactive TV within Sony's broadband strategy. He has led research groups in a variety of companies, including Chorus, HP and Sony, and a variety of countries—the UK, France, Japan, and the USA. He graduated from Lancaster University in the UK. Contact him at the Distributed Systems Lab, Sony Research Labs, 3300 Zanker Rd, MD: SJ2C6, San Jose, CA 95134-1901; rodder@arch.sel.sony.com.



Bob Rau

Embedded systems have been around for a while. Why the sudden change in interest in them? What circumstances were missing in the past that exist now?

Special-purpose systems and embedded systems have, indeed, been around for a long time. Embedded systems were a fairly significant part of Digital Equipment's business even back in the 1970s. Since then, the center of gravity of computing has moved down from expensive systems to less and less expensive ones, and now we are at the point where we can put significant amounts of computing on to a single chip, making an embedded system very inexpensive.

As a result, embedded systems are suddenly impacting many more consumers and products, which is causing a major increase in interest. It's not that embedded systems, or special-purpose architectures, are in some sense new, but they're now becoming so prolific that the interest in them has gone up tremendously.

The ratio of visible computers to people has gone from a thousand people per computer in the mainframe era down to about three or four PCs and laptops per person today. If you extrapolate this forward, we will have to have hundreds of visible computers per person. This makes no sense; instead, almost all of these computers are going to be invisible, embedded computers. Embedded, invisible computers are here to stay, and we will see multiple generations of them just as we did with visible computers.

What will be the biggest challenges for ubiquitous embedded systems?

What features will be crucial for achieving acceptable embedded systems: security, real time, scalability, or high availability?

Security, real time, scalability, and high availability are all important, but it depends on the nature of the application. If it's a smart card, security is important. If it is a life-support system, availability and reliability are clearly important. One could think of examples where any particular combination of these features is important.

You are even talking about feature composability, not just functional composability or object composability.

That's right. The aspects of the computer system that are important differ from one application to the other, but there are some things that I think will be more universally needed. If I come at it from a hardware point of view, the cost and the power that we need at a particular performance level is going to be a big deal as we move into the consumer space. The lower the cost drops, the greater the affordability and the number of users. This combination—of wanting high performance with low cost or low power—is going to be a big challenge.

From a software point of view, a small footprint is important in many cases. Again, it is directly related to the system cost—to less RAM or ROM. There's often no disk in an embedded system; the entire software system has to fit in ROM. A saving of a few cents in ROM cost can be a big difference in a low-priced product.

These hardware and software requirements, such as high performance at very low cost, are pulling in different directions, forcing the entire architecture toward specialization. Specialization often leads to customization, because it is highly unlikely that the special thing you want is sitting on the shelf.

The big challenge then is, if we need to design thousands of smart products and the highly specialized, customized hardware and software systems that go into them, how will we have the aggregate design bandwidth to do it? We have enough difficulty coming out with a few standard systems every year. How are we going to design 10 times or 50 times as many systems a year?

Do you have an answer to this?

Either we've got to come up with a few general-purpose architectures—hardware and software—that can be used off the shelf but that somehow meet these simultaneous, strict demands for cost and performance, or we will have to go the customization route. The former is, I think, unrealistic. In the latter case, the answer is the same as it has always been, automation, to increase the productivity of the designers.

What are the most promising embedded systems applications and the dominant deployment areas? Are there any killer applications that will make or break it?

Clearly, smart products are the killer app. A vast number of very innovative, imaginative, smart products will, on the one hand, be enabled by the existence of low cost, high-performance computing and, on the other hand, will generate the demand. Currently, important areas are communication, both

networking and telecom, imaging and streaming multimedia, digital consumer electronics, and Internet appliances. Robotic products of diverse types will be of increasing importance.

Are standards needed for embedded systems? In which subareas?

We will, of course, need standards for the interfaces between embedded systems that must interoperate, but perhaps none for the stand-alone ones. In all cases, we will be relatively unconstrained in how we design the internals of the embedded systems, enabling specialization and customization.

Who are (will be) the dominant players in this area?

If we judge by history, it will be startups that aren't necessarily prominent today. For example, when you think about the minicomputer industry, the really big names were not the mainframe companies, but completely new companies such as DEC, Data General, and Wang. History seems to suggest that it is the startups that will show the willingness to break from tradition and do interesting things.

Do you think that the innovation will remain within the computer industry or that it may get diffused outside in consumer products?

Innovation is definitely going to happen in many places. There's a collision underway between two industries. Because the entire system fits on a chip, computer companies can't really design systems anymore without becoming system-on-a-chip experts. The semiconductor companies can't design a chip without becoming system designers, because the chip contains the whole system. This is causing quite a lot of confusion in these industries. The two communities that are most focused on the embedded space are the smart product companies, who already have the ability to design embedded systems, and the semiconductor companies that are trying to step up to system-on-a-chip design. These companies are taking the lead.

What is the direction of future embedded operating systems? Will size and the amount of functionality matter in the future? Will OSs be relevant in future embedded systems, or will it become an invisible solution, glued to the application and the rest of the system software?

Again, if we look at history, each time there is a new generation of computing, there's been a new generation of operating systems as well. Linux seems to be quite a force; a lot of companies are addressing the Web space using it.

The fundamental problem that I see is that the operating system community has always been pulled in two directions. One is to provide really compact operating systems that fit on small computer systems. The other is to provide fully functional, sophisticated operating systems that can provide services for every possible application. As a result, operating systems have oscillated back and forth between the two

extremes. It's not at all clear that any one of the existing operating systems is going to be the right one in the embedded space, particularly when there is such an emphasis on specialization.

A question that often comes up has to do with the role of existing operating systems. Right now, it is impossible to fit a standard operating system onto a really small and embedded system. Nevertheless, it might become possible over time, as memory becomes cheaper. In other words, won't it be just a matter of time before standard operating systems dominate the embedded space? I think this is the wrong way to look at it. In fact, what drives industry is the quest for sufficient functionality at the lowest possible price. The difference between these two viewpoints is profound. If one person has the ability to create operating systems that take up 64 Kbytes and another person only has the ability to create operating systems that take up 10 MBytes, the first person will always be able to address a lower price bracket in embedded computing. It might be four or five years before the second person can address that same price bracket. As the price of computing drops, the number of users increases exponentially. So the smart person is not going to wait for the ability to use a large operating system at a certain price bracket. He or she is going to focus on how to go to a low price bracket with a small operating system as early as possible.

Consequently, I don't believe there will ever be a time, in the embedded space, when there isn't a premium attached to having inexpensive hardware and small operating systems. That's not to say that, at some higher price bracket, we might not make use of larger, perhaps standard, operating systems, but we will always be under pressure to minimize cost for some sufficient level of functionality and performance.

How will networking impact embedded systems, in particular, high-speed communication links and wireless?

As the cost of interfacing to the network lowers, the majority of embedded systems will be connected to the Internet, either over a wire or by wireless. As the bandwidth goes up, the transmission of bulky information—images, video, and such—is going to become commonplace. These two trends, along with huge amounts of inexpensive embedded computer power, will result in a very different, and rather exciting, landscape of interconnected smart products and appliances. //

B. Ramakrishna (Bob) Rau is an HPL fellow and the manager of the Compiler and Architecture Research group at Hewlett-Packard Laboratories. He also leads the PICO (Program In, Chip Out) project, which aims to develop the capability to take an embedded application and to automatically design highly customized computing hardware and compilers that are specific to that application. He has been at the forefront of the field of very long instruction word (VLIW) computing since its inception in 1980. His research interests include architecture, compilers, operating systems, and design automation. He received his PhD from Stanford University. He is an IEEE senior member and an ACM member. Contact him at rau@hpl.hp.com.