The Future of Human-Computer Interaction

Personal computing launched with the IBM PC. But *popular computing*—computing for the masses—launched with the modern WIMP (windows, icons, mouse, pointer) interface, which made computers usable by ordinary people. As popular computing has grown, the role of HCI (human-computer interaction) has increased. Most software today is interactive, and code related to the interface is more than half of all code. HCI also has a key role in application design. In a consumer market, a product's success depends on each user's *experience* with it. Unfortunately, great engineering on the back end will be undone by a poor interface, and a good UI can carry a product in spite of weaknesses inside.

More importantly, however, it's not a good idea to separate "the interface" from the rest of the product, since the customer sees the product as one system. Designing "from the interface in" is the state of the art today. So HCI has expanded to encompass "user-centered design,"

which includes everything from needs analysis, concept development, prototyping, and design evolution to support and field evaluation after the product ships. That's not to say that HCI swallows up all of software engineering. But the methods of user-centered design—contextual inquiry, ethnography, qualitative and quantitative evaluation of user behavior—are quite different from those for the rest of computer engineering. So it's important to have someone with those skills involved in all phases of a product's development.

In spite of their unfamiliar content and methods, HCI courses are strongly in demand in university programs and should be part of the core curriculum. At a recent industry advisory board meeting for U.C. Berkeley's computer science division, HCI was unanimously cited as the most important priority for future research and teaching by our industry experts. Ease of use remains a barrier to growth and success in IT even in today's business

JOHN CANNY, UNIVERSITY OF CALIFORNIA, BERKELEY

Is an HCI revolution just around the corner?

The Future of Human-Computer Interaction

markets. And it is surely *the* major challenge for emerging markets such as smart phones, home media appliances, medical devices, and automotive interfaces.

Before we explore the future of HCI, it's important to review some key lessons from the past. Many core ideas in HCI trace back to Vannevar Bush's "memex" paper ("As We May Think," *Atlantic Monthly*, July 1945), J. C. R. Licklider's vision of networked IT as DARPA director in the 1960s, and Douglas Engelbart's amazing NLS (online system) demonstration at the Fall Joint Computer Conference in San Francisco in December 1968. While acknowledging these pioneers, we're going to jump straight to the "modern era" of HCI, which led directly to popular computing. The incubator for this was, not surprisingly, Xerox PARC (Palo Alto Research Center).

THE PAST

In 1970, Alan Kay arrived at the just-formed Xerox PARC inspired by his vision of a laptop computer for ordinary users. Back then, the personal computer was a dream shared by a few wild souls. There were a handful of mini-computers (e.g., the PDP11 appeared in 1970), but those machines were for engineers and scientists, of course. Kay and other PARC engineers (including Butler Lampson and Chuck Thacker) started developing computers with the extraordinary idea of giving them to ordinary people. Kay was also working on Smalltalk (a language for kids), leading to Smalltalk-72 soon after. His laptop-style Dynabook was infeasible in the 1970s, but the group did produce the Xerox Alto desktop computer in 1973. The Alto had a mouse, Ethernet, and an overlapping window display. It was a technical marvel, but not necessarily easy to use. There was mouse functionality, but it was mostly a "text-oriented" machine. It also lacked a killer app (lesson 1). While the Alto was developed for ordinary users, it was not clear at the time what that market really looked like (lesson 2). Most Altos appear to have been sold or given away to engineering labs.

In 1976 Don Massaro from Xerox's office products division pushed ahead a personal computer concept for office environments called the Star. A separate development division was created for the Star and headed by David Liddle. It worked closely with PARC, but was not

part of PARC. The Star is rightfully cited as the first "modern" WIMP computer. It's impossible to look at screenshots, or to actually use a machine (which I was able to do at a retrospective event at Interval Research) without being struck by how good it is compared with what came after. Liddle quipped that Star was "a huge improvement over its successors." It's not just its execution of the WIMP interface and desktop metaphor, but its remarkably clean and consistent "object-orientedness"—right-button menus, controls, and embeddable objects today are a rather clumsy echo of Star's design.

The most remarkable aspect of Star, however, is the *process* its designers used to develop it, which has been widely imitated and which made good interface design a *reproducible* process. Liddle's first step was to review existing development processes with the help of PARC researchers and produce a best-practices document that Star would follow. It included task analysis, scenario development, rapid prototyping, and users' conceptual models. Much of the design evolution happened before any code was written. Code development itself consisted of many small steps with frequent user testing. It was a textbook example (and it's in Terry Winograd's 1996 landmark textbook, *Bringing Design to Software*) of user-centered design.

Even the Alto had followed a much more classical design process. It was enough to put the Alto in the right ballpark, but that machine feels like it's from a completely different era. The Star knew what it was trying to be, and included a good suite of office software. For reasons that almost surely had nothing to do with its interface or application design, it failed in the marketplace. Its close reincarnation in the Macintosh was a huge success. So (lesson 3) good mass-market design requires a user-centered design process. And it often involves real social scientists or usability experts, as well as engineers.

The Star design was so good that HCI researchers are regularly the brunt of "Star backlash." It goes something like this: "HCI hasn't produced major innovations in the last 20 years; the WIMP interface today is almost identical to what it was in the 1980s." In many of the "technical arts," that would be a compliment. In computing, we have 20-year-old artifacts in museums and call them "dinosaurs." But it's wrong to apply that thinking to HCI. Humans are the key element in human-computer interaction. As a species, people don't evolve that fast, and we often take years to learn things well. We have interface conventions in automobiles as well (clockwise means turn right, you drive on the right, and so will I). It's just not good to "innovate" with those. For the time being,

we can't "reflash" people with an upgrade, so let's not go there. The amazing thing is (lesson 4), when you execute the human-centered design process well (in a real usage context, as the Star designers did), you get a design that endures for decades. Multiple generations can learn it and become computer-empowered without worrying about losing that skill later.

For the same reason, when you design something new, it's much better to copy every well-known convention you can find than to make up a new one. As Picasso said, "Good artists borrow from the work of others, great artists steal." So (lesson 5) good HCI design is evolutionary rather than revolutionary.

Finally, there is an overall lesson (number 6) to take away from these two systems. The modern popular computer required two kinds of innovation: free-wheeling, vision-driven engineering, often technology-centered but ideally informed by high-level principles of human behavior (Alto); and careful, context-driven, human-cen-

tered, design evolution (Star). That's a critical point. You need truly creative design and engineering to conceive and execute a radically new idea, but innovation also requires validation. In HCI, validation means that it works well with real users. For that to happen, human-centered design evolution must happen. Innovation in the product is a nice virtue, but it's an option in terms of marketability. Usability is not.

THE PRESENT

It sounds like everything is apples so far. User-centered design works well, we have good office information systems, HCI is a solid discipline (if unexciting because we still like those breakthroughs every few years). So why write an article on the future of HCI, and more to the point, why should you read it? The beef is that IT is not just about office work any more. It's going everywhere (yes, you've heard that, but this time it really is). Because of that, we're due for another revolution (in fact, probably several) in HCI over the next few years.

Let's start with PCs. Where are they now? Intel recently reorganized itself to align with the major market sectors for Intel PCs *today*. Those sectors are office, home, medical, and mobile. That's a lot of PCs in new places, and they're almost all running a Star-style WIMP interface.

What about cellphones? Global cellphone sales are now running at 800 million units per year, about four times the annual sales of PCs (or television sets). Recent years have seen 100 percent annual growth in overall phone sales, and close to 200 percent for smart phones. Sales are nearing saturation in developed countries, but still accelerating in the Third World, which dominates now. Smart-phone sales are about 15 percent of the market now (around 100 million units), but with their faster growth should outnumber PCs by 2008. Smart phones today are about as powerful as a midrange PC from eight years ago, but they waste the latter in media performance. Although only a tiny amount of smart-phone software is around now, it is one of the fastest-growing sectors of the industry. Unfortunately, if you've tried interacting with a nontrivial smart-phone application, you'll know what an ordeal it can be. There has been a brave effort to evolve it from its WIMP interface roots, but it just feels wrong—like a shark in a shopping mall.

A small army of gadgets are fighting for dominance in your living room. If you have a state-of-the-art cable box (which will also record 40 hours of hi-def TV), you know it has all the hardware (but not the software—yet) to connect to any conceivable media device. It has an always-on

MORE

Related articles in ACM's Digital Library:

Davis, M., Smith, M., Canny, J., Good, N., King, S., Janakiraman, R. 2005. Towards context-aware face recognition. In *Proceedings of the 13th Annual ACM International Conference on Multimedia* (November).

Richards, J., Christensen, J. 2004. People in our software. *ACM Queue* 1(10).

Want, R., Pering, T. 2005. System challenges for ubiquitous and pervasive computing. In *Proceedings of the 27th International Conference on Software Engineering* (May).

Yankelovich, N., Lai, J. 1999. Designing speech user interfaces. In *CHI '99 Extended Abstracts on Human Factors in Computing Systems* (May).

These articles will be available online at www.acmqueue.com for an eight-week period beginning July 1. Want full access? Join ACM today at www.acm.org and sign up for the Digital Library.

The Future of Human-Computer Interaction

Internet connection and automatic software upgrades that give it a powerful marketing edge. You'll always get cool new services whether you ask for them or not. Microsoft and Apple have PC-like entries for this market, some high-end TVs include all this in the box, and then of course there are game boxes that pack most of those functions along with super-high-end graphics. I've made myself a guinea pig for this stuff, but it's really a pain to use. The wireless keyboards, cornucopia of remote controls, on-screen letter-of-the-alphabet menus—it's like those early "horseless carriage" steam automobiles that had reins. Once again, something feels really wrong.

The story is similar for the other new markets for IT: medical, automotive, etc. In all cases, we're adapting designs that were beautifully optimized for the office to a completely different environment. If the past is any lesson, that isn't going to work.

THE FUTURE: CONTEXT-AWARENESS

What will work in these new domains? The race is certainly not over, but there are some very good bets. Let's start with the cellphone. It has a tiny screen with tiny awkward buttons and no mouse. From start to finish, it was designed for speech. The microphone and speaker are small but highly evolved, and the mic placement in its normal position is optimal for speech recognition. We'll get to speech interfaces shortly. If it's a smart phone, it probably also has a camera and a Bluetooth radio. It has some kind of position information, ranging from coarse cell tower to highly accurate assisted satellite GPS.

This is all "context" information, in contrast to the "text" you might type on the keyboard or see on the screen. Normally, WIMP interfaces rely entirely on the text you type (let's include mouse input) to figure out what to do. Context-aware interfaces use everything they can. This is particularly relevant to mobile phones. When you're using a phone, you're either in some "place" (café, restaurant, store) where you do rather specific activities, or you're moving between places. If the phone can figure out what that place is, it can also provide services that you want there, or that complement services that that place provides (e.g., song previews in a music store, comparison pricing in a supermarket, stats or replays at a

baseball game). When you're between places, the phone can use other pieces of context to figure out what services to offer, or it can wait for you to ask.

Let's work through a concrete example: It's 7 p.m., it's raining, and you're walking in San Francisco (you're from out of town). You open your phone and it displays three buttons labeled "Dinner?", "Taxi?", and "Rapid transit?". Selecting "Dinner?" will present restaurants you're apt to like (using collaborative filtering) and even dishes that you may want. The other options leverage the fact that the phone "knows" that you aren't driving and that it's raining. It also selects "Rapid transit?" (using that name rather than BART as locals know it, since you're not local), rather than bus or tram options since it knows your destination and/or because BART is easier to figure out for out-of-towners than the MUNI bus and tram system. The system's "smarts" are built on knowledge of other users' behavior, knowledge of your own behavior history and preferences, and the immediate context, which includes time, place, weather, Bluetooth neighborhood, etc. These three pieces represent the three fundamental facets of context that we use in all our work: immediate context; activity context, which is about the history of the particular user and a few others (because many activities are cooperative); and situational context, which is about how other actors typically behave in that situation.

Context-awareness is a dream for marketers. Imagine this: Instead of the user initiating the request for "Dinner?", the phone beeps and presents a message, "Aqua restaurant (a leading San Francisco seafood restaurant) is two blocks away and has a special on salmon-in-parchment for \$20." Now, I'm a very rational person, but I also have a weakness for the pink fish, and when I'm tired and wet and I see that, it really doesn't matter what the other options are. That is an example of a proactive service, which if executed right, should be a boon to both consumers and advertisers. Before you raise the specter of a *Minority Report*-style advertising assault, I should tell you that I don't expect to let just anyone send that kind of message to my phone. I'm going to charge a lot for that (probably in whole dollars), so an advertiser had better be very sure of a conversion before trying it. If so, then I am likely to use that service at that time, and then it's very useful to me. If Aqua restaurant beacons this message to a few seafood-loving out-of-towners in the neighborhood that night and gets two or three conversions, then the restaurant will be ahead. If I get a half-dozen of those in an evening and one of them gives me a good service, then I feel like I've won. If none of them works out, well

then at least I've earned my BART (rapid transit) fare home, and some change.

The technical challenges with making this work well are arbitrarily deep, and many of them do not fall within traditional HCI. They span a large fraction of the scope of Web 2.0 business: rich user history; highly personalized, coupled services; carefully targeted marketing; and social and individual services. It's also absolutely essential to build these systems on a deep understanding of users' behavior, their needs and wants, and the contexts where those services are used, which is where HCI methods come in. It also taps deeply into AI (for user and social modeling and prediction); systems engineering (building and deploying the services); psychology, economics, and other social sciences (for understanding rational and nonrational user behavior); and a very broad notion of security (attacks include "bleeding" advertiser revenue using robots). These challenges are going to engage developers and researchers for decades to come. Since targeted marketing is the source that feeds Web 2.0 companies, improvements here are felt directly (and quickly) on the bottom line. Since there seems to be an arbitrarily deep well for improvements, this is where Web 2.0 companies are going to be putting their attention and resources for a long time.

THE FUTURE: PERCEPTUAL INTERFACES

The other important piece of future interfaces should be "perception." The simplest example is speech recognition, or more accurately, speech-based interfaces. Another example is computer vision. Smart phones are excellent speech platforms, as already noted, but most also have cameras and a respectable amount of CPU power, especially in their digital signal processors. They are more than capable of computer vision using either still images or video from their cameras. A simple example is barcode recognition, which is already available on some camera phones (both 2D and 1D barcode readers have appeared on commercial phones). OCR (optical character recognition) for business-card recognition is also available commercially. Another example is TinyMotion, a phone software application that my lab has developed, which uses the video from a camera phone to compute the phone's motion relative to a background—just as an optical mouse does. This creates a software-only general-purpose 2D mouse for camera phones. TinyMotion is very useful for map browsing (which is why we developed it) in location-based cellphone services. It turned out also to be a nice interface for smart-phone games, which is probably a bigger market than its target.

These niche applications for vision on phones are suggestive, but perhaps not really convincing of the economic value of computer vision for phones. Let's look for a moment at "social media," personal data such as photos and videos that are shared with friends and family. As argued before, the phone is a communicating and social platform, and photo sharing is likely to be one of the most popular uses of multimedia on the phone. With collaborators at Berkeley and in industry, we explored face recognition from camera-phone images. The application is precisely photo-sharing and archival. The user will likely want to share a photo with the people who are in the photo and would like meta-data about who is in the photo so he or she can find it later when looking for specific people. Our results were interesting because we found not only was it possible to recognize subjects reasonably well using computer vision, but also that the recognition accuracy improved significantly when context data was used, as well as computer vision. While our system actually did its recognition on a PC rather than on the phone, we realized that the same state-of-the-art PC algorithms could easily have run on the smart phones

we had used. Computer vision has a big role to play in managing personal media assets, and this reaches into the home, as well as the mobile market.

Turning to ASR (automatic speech recognition) and VUIs (voice user interfaces), we saw a boom in these industries in 2000, followed by a contraction for several years. But 2000 was also the era of wild promises and unrealistic expectations. What should have happened with speech? First of all, when PCs were mostly in offices, VUIs didn't make much sense. Nothing wrong with the technology, but speech is a poor match for most office work. Let's not forget the significant advantages of text for routine business communication: You can scan text for what you want, you can read back and forth if you

The Future of Human-Computer Interaction

don't understand, you can edit text while you're writing it to make sure you say exactly what you mean, and you can forward text through a long chain of readers without losing its meaning. Written text is generally less ambiguous than spoken language that expresses the same meaning—we're not really aware of this, but we're trained from an early age to take more care with text. Furthermore, you can work on text documents without your neighbors listening in. Much knowledge work is about managing structured or semi-structured information (even before computers came along). Most organizations relied on paper to store and move this information around with precision and robustness (again before computers). Speech technology can certainly play a role, but it's wrong to think about displacing most of the "paperwork" in office environments. As Jordan Cohen (formerly of VoiceSignal, now of SRI International) points out in his interview in this issue, the way to succeed with speech technology is first to identify the market where it makes sense.

Let's remember the lessons from the Xerox Star. The Star was all about having a real-use context (office work) and identifying an appropriate set of user tasks. Phones are primarily about communicating using a variety of media (sound, images, text) and to an increasing extent about sharing and archiving those media. To support and augment those communication services, we need some knowledge of what's "in" those media, which is exactly a machine perception task. Furthermore, if phones are to provide other services (besides communication) to users, they also need to interpret the user's intent through whatever interfaces the phone possesses. I already remarked on users' toils with phone menus and buttons, while at the same time the phone is a beautifully evolved speech platform. Speech interfaces do indeed look like a great choice. They continue to improve in performance, but the state of the art is much better than people realize.

Until last year, like most HCI researchers, I was skeptical about the value of speech interfaces in HCI. But then I saw a Samsung phone (P207) shipping with large-vocabulary speech recognition and getting very good user reviews in all kinds of publications (including the hard-to-impress business market).

I also taught a class on medical technologies and had a chance to meet with many caregivers. There is already a large speech industry in medicine, and it is widely seen as one of the key technologies moving forward (it has probably already eclipsed "office ASR" and is a significant part of the speech recognition industry overall).

I had committed the cardinal sin of generalizing experience from a technology in one context (VUIs in the office) to its application in a different context. It's the technology-in-context complex that matters. ASR-on-phones and ASR-in-medicine are brand new markets. Their users don't know or care about the history of speech in the office. They just buy it and use it, and they either like it (so far, so good) or they don't.

My only direct experience with speech interfaces was with the burgeoning automated call-center industry, which had been quite bad. But after learning more about the state of the art (Randy Allen Harris's *Voice Interaction Design* or Blade Kotelly's *The Art and Business of Speech Recognition* are excellent guides), I realized that there are many superb examples of voice interface design. It's a lot like Web sites and GUIs in the 1980s. The practice of human-centered user interface design was not widely known back then, but as the HCI discipline grew both in academia and industry, best practices spread. Products that didn't follow a good user-centered process were quickly displaced by competitors that did. There is an excellent set of user-centered design practices for speech interfaces that are very similar to the practices for core HCI. As yet, they aren't widely adopted, but the differences between systems that follow them and those that don't are so striking that this cannot last forever.

It has also become clear that the recognition accuracy of the ASR part of the interface is not the limiting factor—it's the quality of the overall VUI design and the match of the application to its context. In other words, there's no reason to wait for future technical magic before using speech interfaces. You can write excellent ones now, assuming speech interaction fits your application context. (See the recent examples that appeared in the article "Conversational? Isn't Always What You Think It Is" from *Speech Technology Magazine*, July/August 2003; <http://www.speechtechmag.com>.)

After these epiphanies, I moved a significant amount of activity in my group to speech and dialog-based interfaces (i.e., started four new projects). While there are very good practices in speech interface design today and many useful services that can be built with them, there are still significant challenges and room for improvement. Those limits have to do with the shared understanding between

a human and a machine sharing a speech interface. This is why speech interfaces are also a rich research area. Much of the shared information is the context we have already been talking about, and all of the aforementioned projects are coupled with our work on context-awareness (for more information, see my home page, <http://www.cs.berkeley.edu/~jfc>).

A WORD (OR TWO) ABOUT PRIVACY

Perceptual interfaces imply cameras, microphones, and other sensors capturing the user's behavior. Context-awareness implies high-level interpretation of that data, often in locations remote (in space and time) from where the data was captured. These are all hot buttons for privacy advocates. My group has been working on context-aware systems for eight years, and privacy has always

been an issue. In fact, privacy in ubiquitous computing environments has become a major focus of our group, leading to six papers on the topic. There are a variety of approaches to the problem: better advice and consent interfaces for users, anonymization, and various forms of obfuscation (e.g., reducing the accuracy of location information). I have co-organized workshops on privacy at the Ubiquitous Computing conference for the past four years (UBICOMP 2002-2005), and these have provided a good overview of work in the area (all are available from my home page).

The approach we have taken, and which we are now building into a context-aware prototype, is private computation. In a private computation, user data is cryptographically protected during the computation, and only the final result is revealed. For example, we are interested in the overlap between activities of knowledge workers. It's pos-

sible to infer this overlap by discovering similar keywords in users' e-mails to each other. Normally, doing pattern matching on full e-mail text would be extremely invasive, but the result of the pattern matching is often benign by itself (e.g., if users A and B share a common activity, we typically need only the most salient words or documents related to that activity). Private computation allows us to determine the end result—say, the set of documents related to the activity—without exposing any information at all about the data used to do the pattern matching.

Private computation is challenging to use for a variety of reasons, one of which has been high computational cost. Our most recent result, however, has reduced this by many orders of magnitude and allows privacy to be added to many context algorithms with essentially no computational overhead (accessible as Berkeley Technical Report UCB/EECS-2006-12 from <http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/>). This allows us to compute high-level context information, such as who is involved in an activity and how much (say, as a participation number between 0 and 1) without disclosing when and where the users were actually involved. Private computation provides much stronger privacy protection than anonymization—for example, e-mail with sender/receiver removed (anonymization) is hardly protected at all. Private computation requires some rather exotic techniques (zero-knowledge proofs), but we have built a Java toolkit that is available to others who would like to experiment with it.

CONTEXT-AWARENESS AND PERCEPTION

Context-awareness and perception are really two sides of the same coin. Context-awareness involves interpreting other cues (besides user input) to figure out what a user wants. Many of these cues will require machine perception (is a user talking about food, is there traffic noise, is the sky overcast?). Conversely, machine perception is a difficult task and it "scales" poorly—as you increase the size of the speech vocabulary or the number of potential images to match for vision, accuracy goes down. The task becomes much easier when you add context data to the recognizer. In our research on face recognition, we were able to use available phone context data (time, place, event history) to improve recognition of faces from camera-phone images. In fact, face "recognition" using context data alone (i.e., predicting who's in the image without looking at it) was more accurate than a state-of-the-art face recognizer using computer vision. Putting computer vision and context together, though, does much better than either one alone.

The Future of Human-Computer Interaction

Our work on voice interfaces is attempting to achieve similar gains by adding context data to speech recognition. We think the potential gains are even larger there. But there must be closer coupling between recognizer, the context data, and the application or service built on top of it. That brings us to what is realistically the biggest challenge to contextual and perceptual interfaces: bridging the barriers between the disciplines working on these technologies—specifically, HCI, speech recognition, and computer vision. It's a familiar story when there is a paradigm shift in a technology or market. While there are small communities working on the boundaries, most of the time recognizers are "black boxes" to interface developers. Conversely, folks working on recognition rarely pay attention to context or the applications that come later. We'll make some progress that way, but if we want a revolution, which the market is ready for, then we need to forget tribal allegiances and work together.

OVERVIEW OF THIS ISSUE

The contributions in this issue cover the state of the art in perceptual and context-aware interfaces. In speech interfaces, one of the most exciting pieces of the market is the cellphone. Many cellphones now support speech input for speed dialing or selecting a name from the phone book. Large vocabulary interfaces for dictation appeared last year. Full continuous large-vocabulary recognition is on the way. The latter especially opens up whole new application possibilities for smart phones and may do much to break the usability barrier for these devices. Most of this technology was developed by VoiceSignal. We open this issue with an interview with Jordan Cohen, who recently moved to SRI, but was formerly the CTO of VoiceSignal. Wendy Kellogg, of IBM's Thomas J. Watson Research Lab, and I discuss with Cohen the growth of cellphone speech interfaces, their potential, and the challenges still remaining.

Our second article looks at computer vision-based interfaces. James Crowley, who directs the GRAVIR (Graphics, Vision, and Robotics) laboratory at INRIA (French National Institute for Research in Computer Science and Control) Rhône-Alpes in France, is a leader in this area. A major challenge in high-level interpreta-

tion of human actions is context, as we already noted. Crowley and his colleagues have tackled this problem head-on by developing a rich model of context considering "situations" and "scenarios." This article describes their approach top-down, starting with a representational model and drilling down to their software architecture.

In the third article, we look at context-awareness in a biology lab. Gaetano Borriello, computer science professor at the University of Washington, leads us through some field tests of the Labscape system, which is intended as an efficient but unobtrusive assistant (a Radar O'Reilly) for cell biologists. In this setting, the users' high-level activity is well understood (it's a science experiment). The system has to use available clues from sensing (like most context-aware systems, there is plenty of perception in this one) to figure out where the user is and what resources are needed. Borriello's article is rich with practical advice for making this kind of system succeed.

In our final article, Jim Christensen and colleagues from IBM's Thomas J. Watson Research Lab (including Wendy Kellogg) take a different approach to using context information. Whereas successful automatic context-aware systems are rare at this time, Christensen et al. argue for human interpretation of context information. They describe two systems that exemplify this approach: Grapevine, a system that mediates human-to-human communication to minimize inappropriate interruptions; and Rendezvous, a VoIP conference-calling solution that uses contextual information from corporate resources to enhance the user experience of audio conferencing. They also discuss some cogent issues related to user privacy in context-aware systems. Q

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

JOHN CANNY is the Paul and Stacy Jacobs Distinguished Professor of Engineering at the University of California, Berkeley. His research is in human-computer interaction, with an emphasis on behavior modeling and privacy. He received his Ph.D. in 1987 at the MIT AI Lab. His dissertation on Robot Motion Planning received the ACM dissertation award. He received a Packard Foundation Faculty Fellowship and a Presidential Young Investigator Award. His peer-reviewed publications span robotics, computational geometry, physical simulation, computational algebra, theory and algorithms, information retrieval, HCI and CSCW and cryptography. He has best-paper prizes in several of these areas..

© 2006 ACM 1542-7730/06/0700 \$5.00