IEEE pervasive COMPUTING
MOBILE AND UBIQUITOUS SYSTEMS
www.computer.org/pervasive

Build What You Use

Roy Want

Vol. 5, No. 3
July–September 2006

Editor: Roy Want ■ Intel Research ■ roy.want@intel.com

Build What You Use

Roy Want

To prove success in ubiquitous computing research, you must implement your ideas and deploy them in support of a work practice or community activity. Anything less, and the design's utility will always be doubted, even if the core engineering is outstanding.

The phrase "build what you use and use what you build" was part of Xerox Palo Alto Research Center's culture during the 1980s' heyday of paperless office research. This culture produced the now-ubiquitous personal workstation, local-area network, and bitmapped display. When I joined PARC in the early '90s, building was still very much a part of the culture. With Mark Weiser at the helm of PARC's Computer Science Laboratory, we were encouraged to design ubiquitous computing systems, build them, and learn from them. This turned out to be much more difficult in practice than I first imagined, as we struggled to determine what users wanted and what we should build.

WHAT DO USERS WANT?

Any truly novel capability provided under the banner of ubiquitous computing probably isn't something that users have any experience with, so how do they know they want it?

Photography offers an illustrative example. If, in 1994, you had asked people if they wanted to replace their film camera with a digital version, most would probably have responded negatively. After all, with 35-mm film and inexpensive print shops available in most malls, users could conveniently make high-quality prints with ease. However, by 2004, many people had made the trade, enjoying the ability to print at home, file and sort pictures on their PCs, make photo album CDs, produce slide shows on their TVs, and edit and enhance their images. Until we actually live with a new capability, it's difficult to understand how good or bad it might be.

I've used this example on several occasions to justify the resources necessary for deploying ubiquitous computing research projects to understand what users really want.

TIME TRAVEL

Within a research facility, or for a small user community (approximately 20 people), you can often create a trial system to explore a ubiquitous computing concept in a controlled way with manageable costs. My colleagues sometimes refer to this as "time traveling." By investing in technologies that are prohibitively expensive for a product in today's market and applying them in support of a work practice, we can catch a glimpse of future opportunities.

At Xerox PARC, one of the best examples of time travel was the bitmapped display. When the Alto computer workstation was first created, putting the required number of semiconductor memory chips in a computer's frame buffer to create a bitmapped display would have made the product too expensive. However, the market trend was clear, with memory density following Moore's law. As the new chips became available, the price of the previous, less-dense generation fell. When bitmapped displays eventually became economically viable (no surprise to us today), PARC had carried out all of the important window-and-desktop-metaphor research ahead of any competition.

Of course, turning this advantage into a market leadership position for personal computers was another story, and Xerox has had its own checkered history in the computer business. Predicting the future and knowing what's important is a difficult business to be in. Time traveling in this way doesn't provide all the answers—just some of them.

DESIGNING FOR DEPLOYMENT

It's important to design a prototype system with deployment in mind. Furthermore, the trial user community itself is a resource that you must handle respectfully. The users will almost certainly have things to do other than help you debug your prototype; if the process consumes too much time, you might lose their support. It's also important not to embarrass them—for example, by making them carry around a geeky-looking mobile device. Moreover, now that computing devices have become an integral part of users' lives, you must be careful not to disrupt or significantly alter their experience—for example, by giving

MISSION STATEMENT: *IEEE Pervasive Computing* is a catalyst for advancing research and practice in mobile and ubiquitous computing. It is the premier publishing forum for peer-reviewed articles, industry news, surveys, and tutorials for a broad, multidisciplinary community.

them a prototype cell phone that routinely drops calls.

Going beyond the research concept to a system that lets users believe it might one day be a product requires considerable planning. The underlying system, having been created with limited time and resources, might not have the final product's refined architecture and quality assurance, but you should base the majority of what the trial users see, hear, and touch on the highest standards possible. For instance, providing a stylishly molded case for a prototype mobile device will go a long way toward gaining your users' acceptance.

Plastic molding has traditionally been expensive, but many newer technologies exist that are widely available through companies that specialize in enabling product concepts. Inexpensive molding technologies that can be used to create short production runs of cases include room-temperature vulcanization, stereo lithography, and fused-deposition modeling. But don't expect to go to traditional design houses to find them, as they charge top dollar to create a winning aesthetic for a market. For an unproven concept with a small budget, you'll need to go directly to the prototype shops. Finding a local company that can provide one of these services can be a boost to project deployment, even with a constrained university budget.

I've never found designing an enclosure to be a chore—something I had to complete before performing my real research. It's always been a fun and rewarding part of the project. Adding the 3-mm plastic veneer is the point where a mobile concept comes to life and transitions to a device that could one day become a real product. It's also the point at which you'll likely want to take photographs and share them with your colleagues.

SYSTEM SOFTWARE ISSUES

Even once you take care of a system's user-visible parts, deploying system software for a community can have operational pitfalls.

Although the code might appear functional, you'll need to leave it running far longer than you likely tested during research development. After running for days, memory leaks might cause a system to fail. Also, if the design uses logging, log files might grow to an unmanageable size, slowing everything down or blocking a process from running because the file system is full. There are well-known techniques to deal with both of these issues, but research prototypes often don't plan for this class of problem. You can detect these issues in the lab by simply running long-lived tests.

The real proof of a successful trial deployment is when the users continue to work with a system after the trial is officially over.

Also, users will likely run your system in environments different from your own. They'll also use it alongside other applications, running on the same devices, causing problems related to execution timing. These will be very hard to predict or provoke without an actual system deployment. The only way to really understand this type of fault condition is to build extensive monitoring into the system before it's deployed.

HARDWARE ISSUES

Hardware has its own maintenance challenges. You can always expect to find a few lemons in any batch of prototype devices. Furthermore, some devices might stop working when they get too hot or cold.

More importantly, users will do the most unexpected things. They'll drop your hardware onto solid concrete, inadvertently leave it in garments that end up in the washing machine, and press buttons in the most unlikely combinations, which might freeze the system. Furthermore, mobile devices that were validated with relatively fresh batteries or lab power supplies might start to behave oddly when the supply voltage falls below a critical value, causing some internal components to fail before others.

EXPERIENCE

Needless to say, all these issues make deployments difficult. If you're a veteran of deployments, you'll probably resonate with my comments and add another long list of your own anecdotes. If you're a novice, I hope this gives you a flavor of the problems and encourages you to consult with veterans in your own organization. This will likely save you a lot of time and keep your user community happy. If you sour the users, getting a second chance is usually very difficult. Even though on return you might have largely fixed any problems, a previous bad experience can bias user reactions for the foreseeable future.

Another observation about real products—and why many product debuts seem poorly conceived—is that companies are basically carrying out real-world trial deployments on a mass scale. This approach requires the financial resources of a big company that's prepared to take a large gamble in its attempt to produce the best kind of market data. However, many of the systems we researchers envision are aimed at the medium- to long-range future, and even these companies aren't prepared to take such a high risk.

I conclude with three observations that I hope will be of value to you. First, the real proof of a successful trial deployment is when the users continue to work with a system after the trial is officially over. Second, it's important to deploy your trial system for your own use, and "use what you build" to understand the user feedback. Finally, the key to a successful deployment is to make your users feel smarter than they were before it began. **P**