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

Moving Out of the Lab: Deploying Pervasive Technologies in a Hospital

Thomas Riisgaard Hansen, Jakob E. Bardram, and Mads Soegaard,

University of Aarhus

Vol. 5, No. 3
July–September 2006

Moving Out of the Lab: Deploying Pervasive Technologies in a Hospital

Lessons learned from deploying a hospital scheduling and awareness system can help other researchers prepare pervasive systems for the real world.

Thomas Riisgaard Hansen, Jakob E. Bardram, and Mads Soegaard
University of Aarhus

Research into the nature of ubiquitous and pervasive computing has now been around for more than a decade. However, there are few examples of real-world research-based deployments of pervasive technologies and only a sparse amount of literature about how ubicomp concepts and technologies play out on a large scale.[1,2] This is likely because pervasive technologies are, to a large degree, still premature, and examples of such systems are mostly in controlled lab environments. Another reason is that it's problematic to define a real pervasive computing application or system. Must it include sensors? Should it be context-aware? Embedded? Invisible? And should mobile phones or PDAs be involved?

We can't, and shouldn't, answer these questions. It's more sensible to talk about a system possessing different pervasive computing traits—that is, a system can contain sensors, use large interactive displays, or involve mobile computing technologies. In this sense, researchers can fulfill the application's requirements by inserting pervasive computing technology into the system, which is, simply put, an application that serves some business purpose.

This article reports our experiences with deploying the *iHospital system*, a hospital scheduling and awareness system. We built the system to support the often intense coordination of operations in a large hospital, and to this end, it incorporates location tracking, a context-awareness system, large interactive displays, and mobile phones. We choose not to label our system *pervasive* because, to its users, it's a scheduling, coordination, and awareness system that possesses several pervasive computing traits.

Here we share our experiences in deploying this system at a small Danish hospital, which uses it extensively in the operating ward.

Interactive hospital project

Smoothly, efficiently, and securely running an operating ward requires maintaining a shared overview of the current surgeries and personnel, being able to communicate with the people involved, and being able to quickly adjust to schedule changes. We developed a distributed and heterogeneous system for supporting this line of hectic work. The system, which we developed to support work in the operating ward, also supports work in the related patient and recovery wards.

The system runs on large wall displays, PCs, and mobile phones and uses location-tracking technology and video streaming to provide context information.

Developed technologies

We carried out our entire project in close collaboration with clinicians. They participated in

Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/06/$20.00 © 2006 IEEE

Figure 1. AwareMedia in use in the central coordinating station. Three nurses and a doctor are discussing and updating the schedule.

brainstorming sessions and workshops and have helped refine and evaluate early prototypes. During deployment, we used feedback from the clinicians to refine the iHospital system. The inspiration, design, and implementation process lasted approximately a year and half, and in November 2005, a suite of pervasive and context-aware applications was ready for deployment. The suite consisted of a distributed media space system, a system and infrastructure for providing awareness to mobile devices, and a context-awareness and location-tracking infrastructure:

AwareMedia is an application that shows information about the work in the different operating rooms. A video stream provides overall awareness of a given operation's state, a progress bar shows more detailed information about the progress, a chat area lets people communicate in a less intrusive manner, a schedule shows the current operating schedule including changes, and the location-tracking system shows information about who is in the operating room. This information is distributed to different computers, both within the operating ward and the rest of the hospital. Figure 1 shows the system in use in the central coordinating station.

Figure 2. A doctor can use the AwarePhone client to check the a surgery's status. Here the clinician is running it on a Nokia 6600 mobile phone.

AwarePhone is a program that runs on Symbian mobile phones. It provides an overview of people at work and the status of surgeries in the operating rooms. An augmented phone book lets the user get enhanced presence information about others in the phone book, such as their location, schedule, and self-reported status. With this information, users can decide who to call and when. AwareMedia users can also choose to send a message to an AwarePhone instead of calling. Figure 2 shows the AwarePhone client running on a Nokia 6600 mobile phone.

We used Bluetooth to track people in our *location and context-awareness infrastructure*. All PCs in the operating ward have a modified Bluetooth USB stick with a range just less than 10 meters. The system tracks clinical personnel and patients using a special Bluetooth chip (see figure 3) or their mobile phone. The Java Context Awareness Framework (JCAF) infrastructure[3] handles location, activity, and status information and, in turn, feeds the Aware infrastructure,[4] which provides awareness information for the

Figure 3. Tracking with Bluetooth beacons. The small black item sends a Bluetooth signal to the infrastructure. The chips are carried on the shirt or in the pocket during a work shift and then placed in a charger for the night.

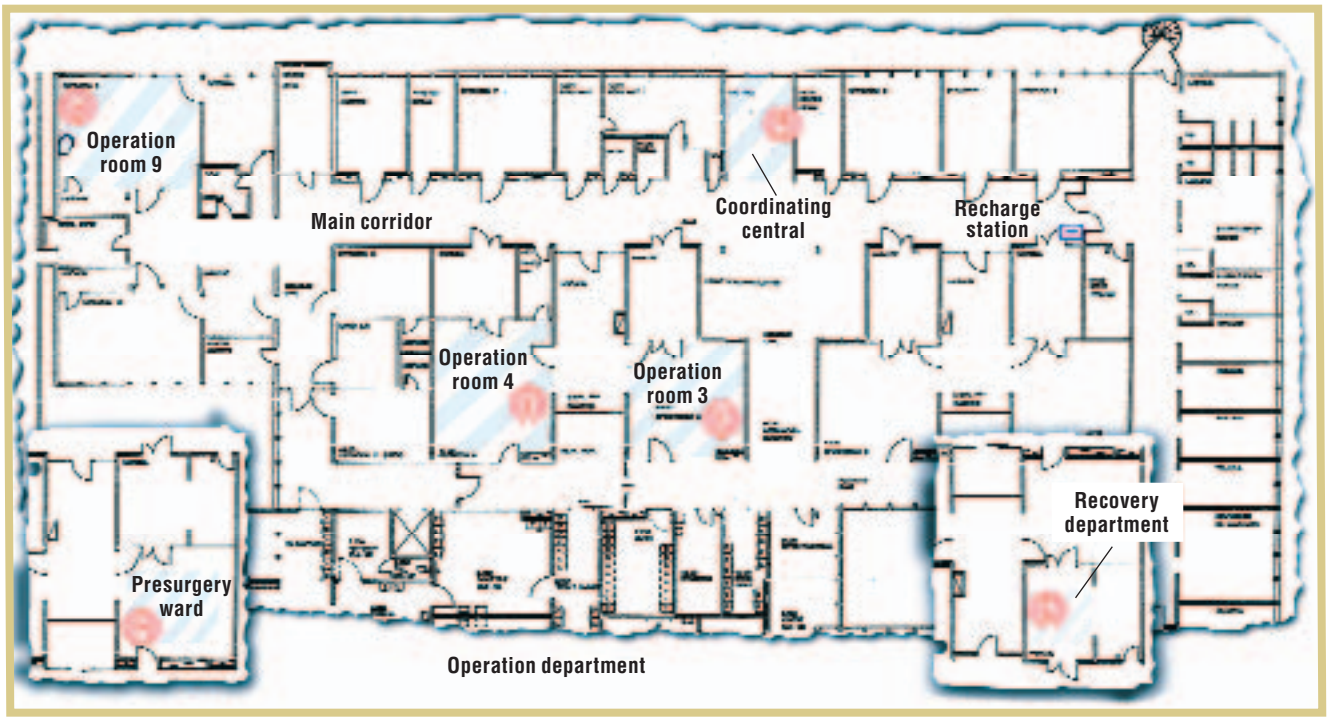

Figure 4. A schematic overview of the presurgery, operating, and recovery wards. Red circles indicate the deployed clients' locations.

AwareMedia and AwarePhone applications the clinicians use.

These systems were developed to support a number of scenarios. This example involves various steps: An acute patient arrives. After a quick glance at a large wall display running the AwareMedia client, the head nurse finds an empty operating room. She touches the screen to schedule a new acute surgery for that operating room. She then uses the location-tracking system to find a surgeon that isn't currently in surgery and sends a message to that person's mobile phone with information about the new surgery. As soon as the nurse enters the patient into the system, the patient ward is notified about the change of plan. This lets the ward inform the next scheduled patient that his or her surgery has been postponed due to the acute surgery. Our setup supports many similar scenarios aimed at providing awareness and enhancing communication.

Deployment

We deployed the iHospital system in November 2005 at Horsens Hospital in Denmark. We installed 10 PCs in the hospital, some in the operating rooms and some in wards and offices in other areas. Two computers are equipped with two 40-inch touch-sensitive displays. All the computers are equipped with a Bluetooth location-tracking system, and the computers in the operation department include webcams. We gave out 17 mobile phones to clinicians, and a central server runs the context-awareness infrastructure and distributes events to the different clients.

Figure 4 shows a schematic overview of the operating ward; the red marks indicate the locations of the deployed clients. The operating ward is on the hospital's second floor, the recovery ward is also on the second floor but approximately 200 meters away, and the presurgery ward is on the fifth floor.

At the time we wrote this article, the system had been deployed for more than four months. In February 2006, the hospital discontinued paper-based schedules and is now using only the computer system. The system handles 200 to 500 operation events per day. An *event* is either a manual creation of a new operation or a manual update to an existing surgery. We handed out 17 mobile phones that the clinicians use in their daily work. We combined the phones with Bluetooth beacons that our system uses to track daily movements for approximately 30 people.

We log interactions with the system and have collected extensive data about its use. With this data, we can see that the surgical staff is using the system extensively, especially the ability to send messages, access video feeds, and check a surgery's progress.

Deployment issues

When deploying the suite of pervasive computing systems we've just described, we faced a range of nontrivial, real-world issues. Some of these were well-known problems in pervasive computing, such as calibrating indoor location systems. Other issues seemed trivial and banal at first, such as finding a place for displays. However, these also turned out to be major issues, and we hope that our "war stories" help others design and deploy pervasive computing technologies.

TABLE 1
A checklist of questions to consider during real-world deployment.

Category	Issues	Questions
Hardware	Cost	What will implementation cost? Will scaling up the system affect the price? Is special equipment needed?
	Security	Is the equipment secure? Is there a risk of theft?
	Environment	Does the environment pose special requirements on the equipment? Is the system going to be used outdoors? Can it handle vandalism? Can it withstand being dropped or cleaned?
	Power	Does the system require a power plug? How long can it run without being recharged? Do the batteries run flat if radio communication is used excessively? How do you recharge the system?
	Network	How will the device communicate? Does it require an Ethernet connection? Is the wireless infrastructure in place? Do you need to transmit data in an external network?
	Space	How much physical space does the system use? Is there space on the wall for large wall displays? Is there table space for another computer? Do the doctors have enough room in their pockets for another device? Is there space on the dashboard for another display?
	Safety issues	Will a system malfunction affect safety issues? What is the contingency plan in case of a full system crash? Will the system interfere with other systems? Can the system pose a threat to the user?
Software	Deployment and updates	How is the software transferred to the device? Does the deployment mechanism scale to a large number of devices? Can you update the system? How do you update the different devices? Are the devices accessible after deployment?
	Debugging	If the system malfunctions, how do you find the error? Does the system store debugging information? How do you detect serious errors in the system? How is logging done?
	Security	Does the system need to be secure? How does it keep information confidential and secure? Is there a concrete security risk?
	Integration	Is the deployed system stand-alone? Does it need to communicate with other deployed systems or integrate with third-party systems? Is there a public API and converters for communicating between systems?
	Performance and scalability	How does the system perform? Is system performance acceptable in the real-world setting? How many devices are needed for deployment? Does the system scale?
	Fault tolerance	What happens when an error occurs? Can the system recover automatically? Can the daily system users bring the system back up to a running state? Is the developer team notified about errors? Is the system configured for remote support?
	Heterogeneity	Does the system run in a heterogeneous environment? Do heterogeneous elements need to communicate?
User setting	Usability	Will end users use the system? If so, how many? Can the average user use the system? Does the interface pose problems? Does the system's overall usability match the average user?
	Learning	How do the users learn to use the system? Is it individual instruction or group lessons? Does the system need superusers? Is a manual or help function needed? How does the user get support?
	Politics	Who controls the system? Does the system change the power balance in the user setting? Who benefits from the system? Is the person that benefits from the system the same as the person that provides data to the system? Does the system require extra work from users?
	Privacy	Does the system reveal private information? What kind of personal information does the system distribute and to whom?
	Adaptation	Is the organization ready for the system? Is there organizational resistance? Will the system change formal or informal structures in the organization?
	Trust	Does the user trust the system? Is the information given to users reliable? Who sends the information?
	Support	Will the developers support the system? Does the support organization have remote access to the system?

We have arranged our discussion of deployment issues around three main categories: hardware, software, and user settings. The hardware category contains a set of core questions that researchers can use to identify some of the issues a pervasive system must address before deploying it in a real-world environment. The software category addresses the state of the software being deployed. Can it handle the challenges of real-world deployment? Is it possible to deploy and update the software, locate errors, and debug the software? Does the system perform and scale? What about security? These questions help identify possible weaknesses in the software when moving it to real-world settings. The user-setting category is a mixture of user and organizational issues, relevant to pervasive systems deployed in a user organization.

Although these categories are broad, we find they adequately capture many of the issues we faced. The checklist in table 1 further details the issues we encountered.

Figure 5. The confined space in a recovery department office made it challenging to place all necessary system hardware.

Getting the infrastructure in place

Striking an appropriate balance between infrastructure cost, security, networking, and so on are often at the heart of designing and deploying hardware and infrastructure for these kinds of technologies. For example, when researchers deployed a large-scale sensor network for habitat monitoring,[5] ensuring a correct balance between the challenges posed by a harsh environment, small spaces (such as a bird's nest), ad hoc networking, and limited power were core design and deployment issues.

In our deployment, most of these issues also had varying impact. Cost is of course always a factor. In any research project, using your funding wisely means cutting your coat according to your cloth. There are many forms of cost associated with a pilot study, but our discussion mainly addresses the costs associated with purchasing equipment. In our project, for example, the cost of a location-tracking system was critical. Commercial, production-strength systems such as Ubisense (see www.ubisense.net) are precise but come with a steep price tag. In a full-scale deployment in a large hospital, these systems come with large costs, especially if the system must track all the patients and clinicians.

The alternative solution—which we chose to pursue—was to develop our own, cheaper, and more coarse-grained location-tracking technology, which also let us use existing mobile phones owned by clinicians or patients for location tracking. This solution worked well insofar as we only required location tracking with a room-level granularity.

Pervasive computing is fundamentally about integration with the physical environment, and another mundane but often overlooked aspect of real-world deployment is space—or rather the lack of it. It's important for designers to keep space limitations in mind. In our deployment, it proved difficult to integrate large amounts of technological devices into already confined spaces. Fitting two 40-inch screens into an operating room wasn't trivial, and placing 19-inch touch screens in the tightly furnished offices around the hospital also proved difficult (see figure 5). Furthermore, getting network and power sockets in the right places became an important issue because safety regulations prohibit cables on the floor of operating rooms.

With respect to power, wireless devices need a place to recharge when not in use. At the hospital, this location had to be both central and secure. A lot of equipment is stolen from hospitals, so finding a secure spot was important. For the location tokens, we used a corner just inside the operating ward (see figure 4). For the mobile phones, we gave each person a charger and the responsibility for charging it in a private place.

When situating new technology inside an existing technical environment, a range of safety issues related to interference, interaction, disturbance, and jamming can arise that researchers must address up front. For example, at our hospital, we needed to investigate if using mobile phones (Global System for Mobile Communication [GSM]) and Bluetooth (2.4-GHz band) could interfere with existing medical equipment running in the operating rooms. For this purpose, we engaged special technicians to test these devices and survey the literature available in these areas. Luckily, they concluded that all the equipment in the hospital had proper shielding. However, this might not be the case in other hospitals.

Installing and launching software

The concerns relating to the software's design, implementation, and testing phases include

- how to deploy the software,
- how to keep it up-to-date,
- how to debug running systems,
- how to integrate different software systems, and
- how to ensure a stable and scalable system that performs adequately.

For example, to ensure stable operation in the habitat monitoring project, researchers needed to install a major system component to monitor the motes' health and status.[5]

In our hospital deployment, these issues also played an important role. After the hardware infrastructure was in place, our next concern was to deploy and continuously update the software. The different parts of the system ran on a server, 10 PCs, and 17 mobile phones. Making regular, often daily, updates posed a challenge. Access to operating rooms is restricted, and to enter the rooms, you must be sterile, which is a time-consuming process of getting dressed in surgical clothes and thoroughly washing. The mobile phones were distributed amongst nurses and physicians and weren't easily accessible for maintenance.

To mitigate these challenges, we developed different semiautomatic and fully automatic updating strategies. For example, AwareMedia queries an update table periodically and automatically installs a new release if one is available. For the AwarePhones, we used a semiautomatic approach that let the user install updates from a default Wireless Application Protocol page.

Integration is often an issue that's deliberately excluded in the creation of proof-of-concepts of new technologies. When entering the real world, however, this issue is of utmost importance and can be the difference between a successful deployment and a failure. On the other hand, systems integration is a cumbersome, tedious task that's often rather costly. Our project faced several types of integration challenges. One was an integration between the hospital's local telephone network and the AwarePhone; our goal was to let users operate AwarePhone like a normal hospital phone. Another was integration between the schedules in AwareMedia with the old mainframe-based scheduling system the hospital used. In this latter case, making this integration was simply too costly (and technically challenging), so we hired a secretary to manually transfer the information during the pilot study. From a long-term perspective, however, researchers can't neglect integration requirements because they are fundamental to the design for deployment in a real-world setting.

Because performance and scalability are clearly important in real-world deployment, developers must accommodate these aspects early in the technology's design. In AwareMedia, for example, we used an IP multicast to stream video between different locations simultaneously. This took up significant bandwidth, which meant we had to create a logical separate network during the deployment. Similarly, we took care to minimize traffic to the mobile phone for scalability and performance reasons. In addition, it's necessary to consider the performance of user interfaces. In AwareMedia, for example, we made sure that only a necessary amount of data was loaded into the client to ensure scalability over time.

In the real world, security and privacy are major issues. For example, our system handles real patient data as well as the medical staff's location information. Within the hospital, the system is shielded by the existing firewall that could be accessed from the outside using a virtual private network. However, to avoid broadcasting sensitive patient and location data across the open GSM network, we had an external service provider install a customized security solution .

On a more general level, we argue that real-world deployment of pervasive computing technologies will often face the challenges of heterogeneity—that is, software systems need to run in a heterogeneous environment involving various networks, protocols, hardware, operating systems, applications, databases, and so on. Hence, design for heterogeneity is fundamental for deployable systems. One strategy we used to mitigate heterogeneity in our project was to design and implement loosely coupled subsystems that can run (and fail) independently of each other. In this way, each subsystem can accommodate and utilize the peculiarities of different platforms, networks, and languages and still work together as one system. Changing or updating one component doesn't affect the entire system, and a breakdown in one component only affects part of the system's functionality. For example, a breakdown in the location-tracking system only affects the display of location information. Moreover, debugging a loosely coupled component is easier, and we were able to handle scalability issues for each subsystem. Finally, a loosely coupled system let us use different programming languages depending on the challenge the particular subsystem presents.

Because performance and scalability are clearly important in real-world deployment, developers must accommodate these aspects early in the technology's design.

A related technical strategy mitigating the challenges of robustness, fault tolerance, and performance is to use stateless components. In case of a serious error, it's possible to restart and reinitialize stateless components with no loss of state information. Ensuring stability by restarting stateless processes

Figure 6. Video streaming from an operating room.

has been a simple but effective strategy in embedded software engineering,[6] and these principles are applicable to a high degree in a pervasive computing environment. In our project, we can restart all subsystems independently, including AwareMedia, which we can restart by pressing an on/off button. The only stateful component is the database on the server; all other components initialize from the server. This architecture exposes a single point of failure, but it has the advantage that all other components can fail and restart without losing vital information.

Involving the users

The last group of issues addresses the user setting in which you plan to deploy the developed system. Because our project was designing an end-user system, we focused on usability from the outset. We wanted to deliver a walk-up-and-use system that required no prior training. All objects are visually represented, and users interact through direct manipulation—dragging, dropping, or otherwise visually manipulating most objects.

One special challenge in a hospital deployment is to teach a large number of users, who work in shifts, how to use the system. To this end, we employed a rumor-based and guerrilla-style teaching strategy. We encouraged people to pass on the information we gave them, experiment with the system, and use the large wall displays to watch how other people used the system. Also, we assigned five *superusers* to be the main "rumor spreaders." As a supplement to rumor-based learning, we used what we call guerrilla-style teaching, where we stopped random passersby and taught them how to use new additions to the system.

A completely automated system (such as a context-aware system) can provide valuable information to users without any user interaction. Video streams and location tracking provides our users with valuable information without requiring any user interaction. However, automated systems are seldom enough, and if users are required to take action, the question about who benefits from the system arises. Is it the users, researchers, managers, or colleagues? Do the pervasive components require extra work from users? In our project, it became a problem that users without a mobile phone needed to pick up a Bluetooth chip from the charging station and register it in the system every morning to be part of the tracking system—extra work that mainly benefited their colleagues. The result was that people tended not to pick up a chip, and we didn't find an easy solution to this problem.

Distributed sensors collecting information about the environment and human activities are often part of pervasive systems, introducing privacy issues. For example, based on a study in an elder-care facility, Richard Beckwith discusses how ubiquitous technologies affect privacy.[7] Our system also contains many privacy-sensitive components. Video multicast of work in operating rooms (see figure 6) and location tracking are two examples. Based on previous research on this topic, we expected privacy to be a major problem in the deployment. However, we have been surprised to find how little privacy concerned our participants. Through interviews and observation, we concluded that users trusted our system because of an overall well-functioning work environment and some well-chosen design decisions, such as using only partial location tracking and low-resolution video streams. For example, our location-tracking system doesn't track people in all locations, providing "tracking free" areas such as the coffee room, cafeteria, and bathrooms.

An interesting aspect of trying pervasive technologies out in an organization is to study the adaptation of new technologies. System adaptation is a process where the technology is either integrated into daily work, marginalized as something only a few enthusiasts use, or even completely abandoned. To facilitate system adaptation, we chose an iterative deployment. During each iteration, we deployed and tested a small part of the system, addressing smaller problems as they arose. It took approximately a month to deploy the full-scale system. Using the iterative

deployment strategy, we were able to catch serious organizational or user inconsistencies in the system early on.

A final and often forgotten issue is to consider who is responsible for supporting the system after it's deployed. Supporting a pervasive system can be both time-consuming and challenging because its components might be distributed over large areas. Well-designed software architecture can reduce, but not remove, the need for a support solution. Deciding whether the developers or the organization in which the system is deployed will provide support constitutes an important part of deployment considerations. In our project, the system must run during the daytime and preferably 24 hours a day, seven days a week. We formulated a shared support agreement for the project where the hospital performs smaller support tasks and people from the developing team at the University of Aarhus carried out larger support tasks.

Technologies used in pervasive systems are often under development or implemented as a prototype. If novel pervasive systems are going to be tested in real-world settings, a research prototype isn't enough. With the issues we address in this article, we hope to have provided a useful tool for testing and discussing real-world issues with pervasive systems before and during deployment.

Large-scale deployments of pervasive systems are still rare, but we expect that researchers will test more systems in real-world environments with a nontrivial number of users. We have had few projects to learn from and were forced to learn many lessons by trial and error. As we've shown, issues that seem trivial in the laboratory might become major obstacles when deploying systems in real-world settings.

the AUTHORS

Thomas Riisgaard Hansen is a doctoral student in computer science at the University of Aarhus. His research interests include human-computer interaction with pervasive technologies, interacting with mobile devices, gesture and speech interaction, and technology for the medical domain. He received his MS in information studies from the University of Aarhus. Contact him at Åbogade 34, 8200 Århus N, Denmark; thomasr@daimi.au.dk.

Jakob E. Bardram is an associated professor at the Department of Computer Science at the University of Aarhus. His research interests include pervasive computing, software architecture, computer-supported cooperative work, and applying software within healthcare. He received his PhD in computer science from the University of Aarhus. Contact him at the Computer Science Dept., Univ. of Aarhus, Åbogade 34, 8200 Århus N, Denmark; bardram@daimi.au.dk.

Mads Soegaard is a doctoral student in computer science at the University of Aarhus. He conducts his research in designing interaction for computer-supported collaborative work in ubiquitous computing environments. He received his master's degree in information studies from the University of Aarhus. Contact him at Åbogade 34, 8200 Århus N, Denmark; madss@daimi.au.dk.

ACKNOWLEDGMENTS

We thank our project group from Horsens Hospital, especially Marie Louise Ulsøe, Lisbeth Meier, Jan Bjørn Nielsen, Ole Glerup, Jens Ole Storm, and Steen Friberg. This work has been financially supported by the Competence Centre ISIS Katrinebjerg project 107. We also thank Christian Jonigkeit and Martin Mogensen for their technical support and development.

REFERENCES

1. N. Davis and H.W. Gellersen, "Beyond Prototypes: Challenges in Deploying Ubiquitous Systems," *IEEE Pervasive Computing*, vol. 1, no. 1, 2002, pp. 26–35.
2. J. Scholtz and S. Consolvo, "Toward a Framework for Evaluating Ubiquitous Computing Applications," *IEEE Pervasive Computing*, vol. 3, no. 2, 2004, pp. 82–88.
3. J.E. Bardram, "The Java Context Awareness Framework (JCAF): A Service Infrastructure and Programming Framework for Context-Aware Applications," *Proc. 3rd Int'l Conf. Pervasive Computing* (Pervasive 05), Springer, 2005, pp. 98–116.
4. J.E. Bardram and T.R. Hansen, "The AWARE Architecture: Supporting Context Mediated Social Awareness in Mobile Cooperation," *Proc. 2004 ACM Conf. Computer Supported Cooperative Work* (CSCW 04), ACM Press, 2004, pp. 192–201.
5. A. Mainwaring et al., "Wireless Sensor Networks for Habitat Monitoring," *Proc. 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications*, ACM Press, 2002, pp. 88–97.
6. J. Armstrong et al., *Concurrent Programming in Erlang*, Prentice-Hall, 1996.
7. R. Beckwith, "Designing for Ubiquity: The Perception of Privacy," *IEEE Pervasive Computing*, vol. 2, no. 2, 2003, pp. 40–46.