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

Pervasive Computing Support for Hospitals: An Overview of the Activity-Based Computing Project

Jakob E. Bardram and Henrik B. Christensen

Vol. 6, No. 1
January–March 2007

Pervasive Computing Support for Hospitals:

An Overview of the Activity-Based Computing Project

The activity-based computing project researched pervasive computing support for clinical hospital work. Such technologies have potential for supporting the mobile, collaborative, and disruptive use of heterogeneous embedded devices in a hospital.

Jakob E. Bardram
IT University of Copenhagen

Henrik B. Christensen
University of Aarhus

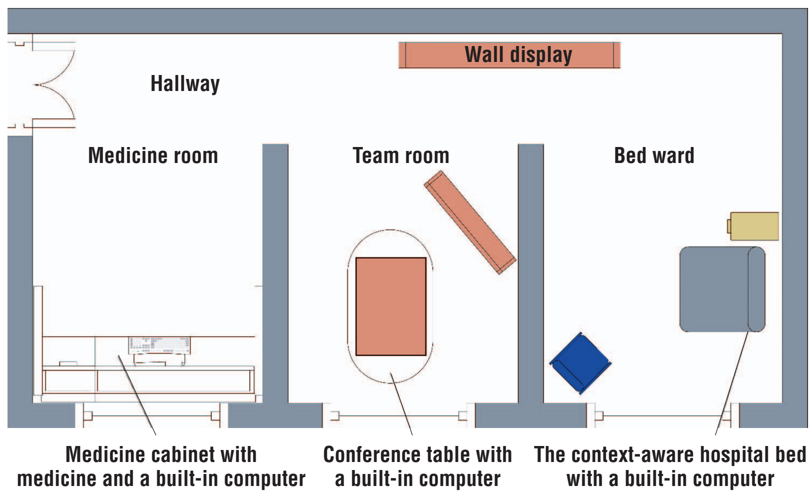
A professional hospital staff's work is challenging to pervasive computing researchers in several ways. Clinicians must handle large volumes of shared data such as patient records and x-rays. Their work is team oriented, with much collaboration between different fields of expertise. It's nomadic because treatment involves talking to patients, attending conferences, and conferring with colleagues—and physicians and nurses don't usually have desks. Their environment is hectic and filled with disruptions, so they must memorize several parallel pending activities. And they need quick access to relevant data for alternating work situations while keeping sensitive medical data private.

However, even though hardware has seen many breakthroughs—making computers small, wearable, and mobile—the software infrastructure has evolved surprisingly little. Operating systems and applications running on small devices are basically scaled-down implementations of desktop software that supports stationary, noncollaborative work on private data in an office setting—that is, characteristics that are directly opposite of those that distinguish the reality of clinicians' work. This put our research group in an odd position. We could demonstrate all sorts of wonderful devices for clinicians—wall displays for large x-rays and bulky information networked with PDAs—but the available software infrastructure hindered our visions of software that would ease secure data access, colleague collaborations, and pending and parallel task management.

As an illustration, consider a physician trying to find a diagnosis for a patient. Our field work showed that this process is typically lengthy and incremental, gathering bits of information in many different locations. For example, at the patient's bedside, the physician enters notes in the electronic patient record (EPR). During the radiology conference, the physician studies x-ray images with a radiologist. At the morning conference, the physician discusses proper medication with colleagues while browsing medicine catalogs. Later, the lab releases a blood sample result, and the physician must study it with a colleague. So, the physician must have access to many applications that show the proper data in different locations, while still tending to other activities and usually sharing material with colleagues. Prevailing operating systems introduce a lot of overhead because the physician must constantly log in and out of devices at hand, starting and stopping sets of applications,

 Published by the IEEE Computer Society ■ 1536-1268/07/$25.00 © 2007 IEEE

Figure 1. The layout of our "Future Hospital" evaluation facility configured to resemble a hospital ward. The ward contained a medicine room, a team conference room, a bed ward, and a hallway.



and browsing each to present the proper view for alternating activities.

Our response to these challenges has been *activity-based computing*, which lets users organize their handling of devices, services, and data in terms of computational activities that facilitate recreating, sharing, and swiftly switching computational context on demand at whatever device is at hand.

Research process

ABC's principles result from a user-centered design process. It began in 2001 as a joint project between a major Danish EPR vendor, a large university hospital, and the Centre for Pervasive Healthcare. Building on the ABC project's long history, the current design principles and technology crystallize solutions for numerous fundamental challenges in designing computer technology for medical work in hospitals.

The ABC project has followed an iterative design process, addressing five medical themes that reflect major working areas in large hospitals: medicine administration by nurses, medicine prescription by physicians, collaboration between clinicians, medical conferences, and surgery. Each theme has engaged in the following research activities:

- extensive field studies of medical work in hospitals, which identify core aspects of medical work and existing technology use;
- a vision workshop for problem identification, idea generation, and focus selection;
- a design workshop for moving toward a specific design of pervasive computing technology;
- a prototype implementation of the proposed pervasive computing technology; and
- an evaluation workshop at which the research prototype was presented to the clinicians, who then used and evaluated it.

In all stages, clinicians have been heavily involved: more than 20 clinicians (including nurses, physicians, surgeons, anesthesiologists, radiologists, and pharmacists) have participated in the project, and we've conducted more than 15 day-long workshops for evaluating our design and technology. Clinicians who had never been introduced to ABC and had not seen our framework participated in four of these evaluation workshops.

Each workshop lasted approximately six hours. We applied scenario-based evaluation methods, asking clinicians to role-play several clinical scenarios. We videotaped and later analyzed all workshops. The evaluation illustrated how an EPR would look if built using the ABC framework. This included EPR applications such as

- a medicine schema, which provided an overview of a patient's prescribed medicine and contained tools for prescribing and administering medicine;
- medical data charts, which contained charts on a patient's blood pressure, pulse, temperature, and weight; and
- an x-ray viewer, which showed an overview of a patient's most recent radiology images.

Figure 1 shows the test facilities' basic layout (the layout varied from workshop to workshop, but figure 1 illustrates the basic components in our test setup). We modeled a large room as a hospital ward with a bed room, a medicine room, a small team conference room, and a hallway connecting these rooms. In addition, we used a remote room (not shown) as the radiology conference room. Figure 2 shows pictures taken during the evaluation sessions.

Reflecting on this design process, the most striking observation is that the five themes didn't result in five singular solutions addressing the challenges within each theme. Instead, proposals gradu-

Figure 2. Evaluation workshops included (a) activity roaming and discovery during patient medication, (b) a radiologist in a synchronous sharing activity, and (c) a clinician roaming the activity to the wall display while engaging in activity sharing.

ally merged into a coherent conceptual and technical framework that successfully addressed the range of challenges simultaneously. We think that one of the ABC project's main contributions is that the ABC concepts and technical principles constitute a coherent conceptual framework for a pervasive computing platform, which supports a wide range of pervasive computing aspects, ranging from mobility to collaboration to context-aware computing.

Other projects have explored aspects of our principles (see the Related Work sidebar), but to the best of our knowledge, none has intensively tested them involving people outside the research groups or domain experts in realistic settings.

Activity-based computing principles

Our work within the five themes lead to a number of principles that underlie ABC.

Activity centered

Clinicians must handle a large amount of data that's often tied to specific work activities. However, when studying how clinicians are using computers, different computer applications often support different parts of an activity. For example, a picture, archiving, and communication system (PACS) supports the application for viewing x-ray images, the medicine schema is shown as part of an EPR, and ordering blood tests is part of a booking and scheduling system. Although all these applications support the same human activity, such as diagnosing a patient, there's little support for aggregating related sets of applications, services, and data into the logic bundle associated with this activity. Consequently, clinicians spend considerable time starting applications, finding proper data, browsing, and navigating user interfaces to present all relevant information.

The principle of activity-centered computing suggests that computational activities should be a first-class object in the computing environment along with, for example, files and applications. A computational activity ("activity" for short) bundles a coherent set of applications, their associated data and resources, and the user interface state needed to support a specific human activity. Figure 3 illustrates how an activity groups together related services and applications and relevant medical data.

An activity-centered computing platform significantly reduces the users' overhead of starting applications, data browsing, and user interface navigation: the activity embodies all this information, and when the users select it, the activity-based platform performs all this setup. Because many types of clinical activities are recurring, support for reusing activities, cloning them, or using activity templates helps clinicians set up relevant digital working environments that contain the needed services and readily accessible data.

In our evaluations of the ABC technology, users appreciated the support for aggregating related resources into activities. Collecting patient-related information into one activity had an immediate benefit and was clearly useful. But the support for creating other activities across a set of patients—for example, a radiology conference activity containing data from patients with the same problem—was also useful. In most EPR systems, all interaction centers on the patient, making it difficult to bundle data for multiple patients in, for example, a conference situation. ABC's activity concept doesn't include semantics, so the user is free to bundle any applications and resources in an activity. We call such activities lightweight because they aren't semantically restricted to, for example, one patient.

This lightweight property was, however, a challenging part of ABC. Because an activity carries no semantics, it was often hard for clinicians to distinguish activities. A recurrent observation was that one activity merged into another—there was no clear demarcation of where one activity (for example, prescribing medicine for patient A) ended and where another (looking at medical data for patient B) began. This problem surfaced when a physician would perform an activity labeled "prescription for Mr. Hansen" and then also look up medical data for Ms. Pedersen. This caused much confusion because our prototype's activity browser showed the name "Mr. Hansen" when the underlying data in the EPR concerned "Ms. Pedersen." Furthermore, when should you create a new activity—when doing the same

Related Work in Activity Support

Several other projects and systems have explored aspects or principles similar to those developed in activity-based computing. We distinguish between two types of related work:

- projects that researched the activity-centered approach and
- projects that researched architectural, technical, and user-oriented aspects relevant for activity-based computing.

The first group is rather small. The most comparable work is Project Aura,[1] which is similar to ABC in its research question and proposed architecture. It also explores the activity concept (denoted tasks), suspend-resume, and roaming. However, Aura hasn't reported discovery results or explored the collaboration aspect that's vital in hospitals and other domains. Furthermore, ABC has tested the principles in practical real-life settings, while Aura has focused on technical evaluations.

The IBM Unified Activity Management project proposes an activity as an explicit computation construct supported by an infrastructure.[2] UAM's activity concept embodies human intent and purpose, which works as a semantic glue between users' tasks and computational objects such as email, calendar entries, chats, WWW resources, and so forth. In comparison, the ABC approach is more lightweight; a computational activity only bundles applications, leaving users to define the bundle's semantics. Domain-specific semantics is useful, however, and it would be interesting to combine the approaches.

Numerous projects in the second group have explored issues similar to those arising in ABC. ABC's central principles are activity suspend-resume and roaming, which entail state preservation and migration between devices. Similar techniques are central in process migration and have been explored in various contexts, including operating systems, programming languages, and agent systems.[3] Typically, the process state is device dependent (for example, on memory contents or the processor state) and might use a binary image as a snapshot. Although agent systems such as Masif[4] and MetaGlue[5] allow agent migration over heterogeneous devices, they still rely on a comparable execution environment such as a Java virtual machines or Corba middleware. ABC, in contrast, represents state in a form independent of the operating system and application details.

Different research projects have presented approaches to seamlessly switch between different sets of applications. Rooms[6] was the first to introduce virtual desktops, allowing users to arrange applications in "rooms" and easily switch between them. GroupBar[7] and Task Gallery[8] are recent examples in this tradition, where "tasks" are defined as sets of applications. However, these systems are mainly window managers, so they don't support adaptation, roaming, sharing, or discovery. The SunRay system (http://sun.com/sunray) supports roaming of running X Window sessions running on a server, but ABC adds support for persistence, collaboration, and adaptation over heterogeneous devices.

Finally, medical-informatics research has recognized the need for making medical applications that are aware of clinicians' tasks—for example, as workflow support systems[9] or clinical guideline systems.[10] However, these systems are clinical applications supporting the flow of medical work and, as such, aren't basic middleware support for pervasive computing.

REFERENCES

1. D. Garlan et al., "Project Aura: Toward Distraction-Free Pervasive Computing," *IEEE Pervasive Computing*, vol. 1, no. 2, 2002, pp. 22–31.
2. T. Moran, "Unified Activity Management: Explicitly Representing Activity in Work-Support Systems," *Ecscw 2005 Workshop Activity: From a Theoretical to a Computational Construct*, 2005; www.daimi.au.dk/~bardram/ecscw2005.
3. D.S. Milojicic et al., "Process Migration," *ACM Computing Surveys*, vol. 32, no. 3, 2000, pp. 241–299.
4. D. Milojicic et al., "Masif: The OMG Mobile Agent System Interoperability Facility," *Personal Technologies*, vol. 2, no. 2, 1998, pp. 117–129.
5. M. Coen et al., "Meeting the Computational Needs of Intelligent Environments: The MetaGlue System," *Proc. 1st Int'l Workshop Managing Interactions in Smart Environments* (Manse 99), Springer, 1999, pp. 201–212.
6. J.A. Henderson and S. Card, "Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface," *ACM Trans. Graphics*, vol. 5, no. 3, 1986, pp. 211–243.
7. G. Smith et al., "GroupBar: The TaskBar Evolved," *Proc. Ozchi 2003*, 2003, pp. 34–43.
8. G. Robertson et al., "The Task Gallery: A 3D Window Manager," *Proc. Sigchi Conf. Human Factors in Computing Systems* (CHI 00), ACM Press, 2000, pp. 494–501.
9. F. Malamateniou and G. Vassilacopoulos, "Developing a Virtual Patient Record Using XML and Web-Based Workflow Technologies," *Int'l J. Medical Informatics*, vol. 70, July 2003, pp. 131–139.
10. P. Ciccarese et al., "Architectures and Tools for Innovative Health Information Systems: The Guide Project," *Int'l J. Medical Informatics*, vol. 74, July 2005, pp. 553–562.

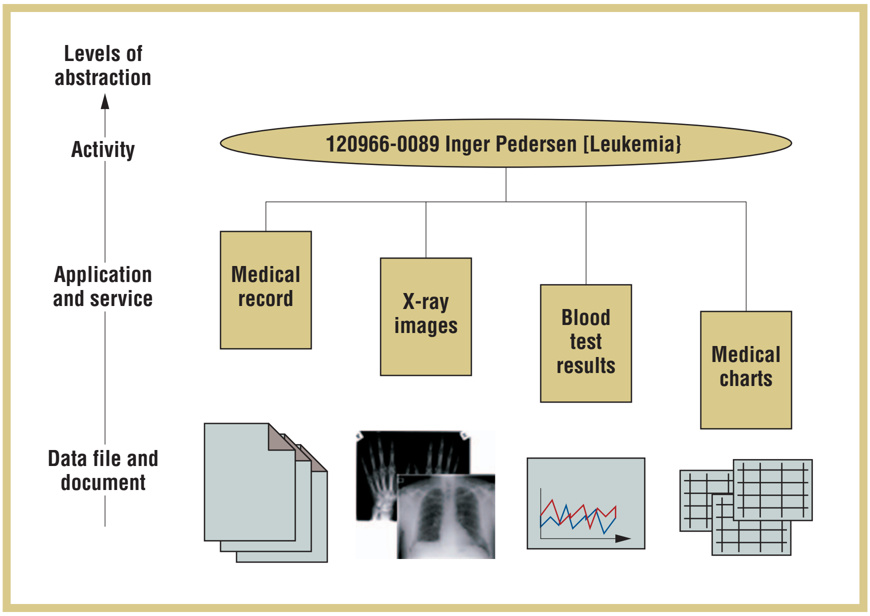


Figure 3. The activity labeled "120966-0089 Inger Pedersen [Leukemia]" is used to bundle a set of services and associated data relevant for Ms. Pedersen's leukemia treatment.

activity (for example, writing prescriptions) for a new patient or when shifting to a new activity altogether (for example, from writing prescriptions to dispersing medicine)?

In general, the cost of using activity-centered computing is the need for the user to create and manage activities. Based on observations during the workshops, different solutions for lowering this overhead emerged, including support for activity templates for frequently used activities and a post hoc bookmark of an important place in the unfolding of activities. For example, if the physician had finished caring for Ms. Pedersen, he could bookmark the activity and later recollect his treatment of her in his office. These solutions gradually merged into the overall principle of activity discovery that's designed to mitigate the cost associated with handling activities.

Activity discovery

This principle suggests that the computing infrastructure should help users identify, create, and manage activities in everyday work. On the basis of recurring activity heuristics and context information regarding location, and by using prototypical activity templates, the infrastructure should be able to hypothesize what activities are going on and prepare relevant computational resources for clinicians' use. Experiments have tested both procedural and logic-programming techniques to handle activity discovery.[1] For example, by combining knowledge about a medicine-administration activity template, the patient's identity, and the medicine tray closest to a nurse (as reported by a location-and-context-awareness system), ABC's infrastructure can retrieve and display relevant medical information and services for the nurse's use. Similarly, in our current use of ABC technology in operating rooms, the technology's infrastructure creates activities on the basis of knowledge about scheduled operations. Such activities contain links to the patient's medical record and recording equipment in the room (that is, to data and services that the surgeon might need during the operation).

Activity discovery's main benefit is that it lowers the overhead of creating activities by supporting semicreation of relevant activities for a specific work situation. The main drawback of these principles is the possibility of creating trivial or meaningless activities that could distract or confuse the user rather than help. During the evaluation, clinicians deemed activity discovery useful in certain situations, especially where there's a simple relationship between some real-world context information and a relevant activity. This typically applied to nurses' work, such as dispensing medicine. Activity discovery was less obvious in other cases, such as creating activities to support outpatient treatment. This couldn't wait until the patient was at the physician's office but had to be created earlier to prepare the physician for the consultation. Relevant context information doesn't just include the physical context but also the digital context (that is, the daily schedule of outpatient consultations can reveal relevant activities for the physician). We use this approach in our current activity discovery for supporting surgical operations.

Activity suspend-resume

Hospital physicians are typically responsible for the treatment of 10 to 15 in-hospital patients and for 100 or more patients in the outpatient clinic. In addition, they often have research, teaching, and administrative responsibilities. So, physicians juggle many concurrent activities that require different levels of attention during a day. And because all hospital work is highly collaborative, interruptions are frequent.

The activity suspend-resume principle suggests that users should be able to quickly switch between activities. So, while working on activity A, the user can select activity B, reconfiguring the computing device with B's applications, data, and user interface state. This suspends activity A, which is then dormant until later resumed.

The benefit is that the platform supports many parallel activities, largely avoiding interruptions in the workflow due to the overhead of starting an application and browsing the user interface. Selecting any pending activity does this automatically. As such, support for activity suspend-resume is somewhat similar to virtual desktops as pioneered by Rooms.[2] At the user interface level in our ABC prototypes, we experimented with a real task bar that switches between activities and not applications (as is the case in Windows).

Our evaluations of the ABC technology clearly showed that support for handling multiple parallel activities using suspend-resume in a medical setting was useful. The clinicians typically created one activity for each patient, and they frequently suspended and later resumed such activities during the workshops. Activities were also saved across workshops so that the users could keep them throughout the evaluation period. The support for saving the user interface's exact state was deemed crucial in the support for fast switching between activities; when users resumed an activity, they would return to the same look-and-feel as when they left the activity.

Activity roaming

Clinical work is nomadic. A typical working day for the average physician or nurse involves walking several kilometers—for one nurse that we followed during our field work, up to 14 km during a shift. Clinicians' activities aren't tied to a specific location but often involve a few small steps in many different places.

The principle of activity roaming supports nomadic and mobile computing by letting clinicians resume computational activities on arbitrary devices, including embedded and handheld devices.

By decoupling computational activities from computational devices, we essentially decouple the clinician's activities from specific locations in the hospital; for example, the physician can resume writing a prescription anywhere if he or she has access to a device. We used activity roaming consistently during the evaluation workshops, and clinicians appreciated the ability to suspend work in one location and resume it later in another. For example, a nurse would walk back and forth between the medicine room and the bed ward (see figure 2a) numerous times each day, so having activity roaming in these places was essential. Physicians who roam around the entire hospital found the activity-roaming support even more useful. We found that activity roaming is a fundamental requirement in creating efficient hospital information systems.

Although our technology lets users roam activities to a PDA or a mobile phone, this didn't make much sense because these devices provide little or no view on medical data. So, in the evaluation workshops, we used larger displays (at least 1,024 × 768 pixels), including tablet PCs, laptops, desktop PCs, and wall displays (see figure 2c). Earlier versions of the ABC technology had no support for adapting the activity to different screen resolutions, but we addressed this problem in later versions with an activity-zooming feature.[3]

Medical work in general, and in hospitals in particular, is highly collaborative due to the specialized nature of medical treatment.

Activity sharing

Medical work in general, and in hospitals in particular, is highly collaborative owing to the specialized nature of medical treatment. Treatment and care are inherently distributed among different medical doctors, nurses, and care assistants, who need to collaborate across time, space, and organizational boundaries.

The principle of activity sharing states that an activity lists participants who can all access and use it. Activities can be shared asynchronously when two or more participants take turns in resuming the activity. Here users can take over the activity from others and see what has been done so far. A typical example is handing over some activity across shifts. An activity can also be shared synchronously when two or more participants resume the activity simultaneously on different computers (resulting in close collaboration among participants working from different locations). A typical example is a radiologist and physician discussing diagnosis and treatment while examining x-ray images.

At the technical level, activity sharing is achieved by reusing the state management mechanisms used in activity suspend-resume.[4] Asynchronous activity sharing is equivalent to activity suspend-resume and activity roaming for more than one user. In synchronous activity sharing, the prototype system distributes state information among the online participants. In addition, it provides different collaborative widgets that support synchronous activity sharing across remote computers, including telepointers, a voice link between participants, and a window showing the activity's participants.

Figures 2b and 2c show the evaluation of real-time activity sharing. In general, clinicians appreciated the system's native support for activity sharing. They also considered support for distributed, real-time activity sharing between, for example, a radiologist in the radiology

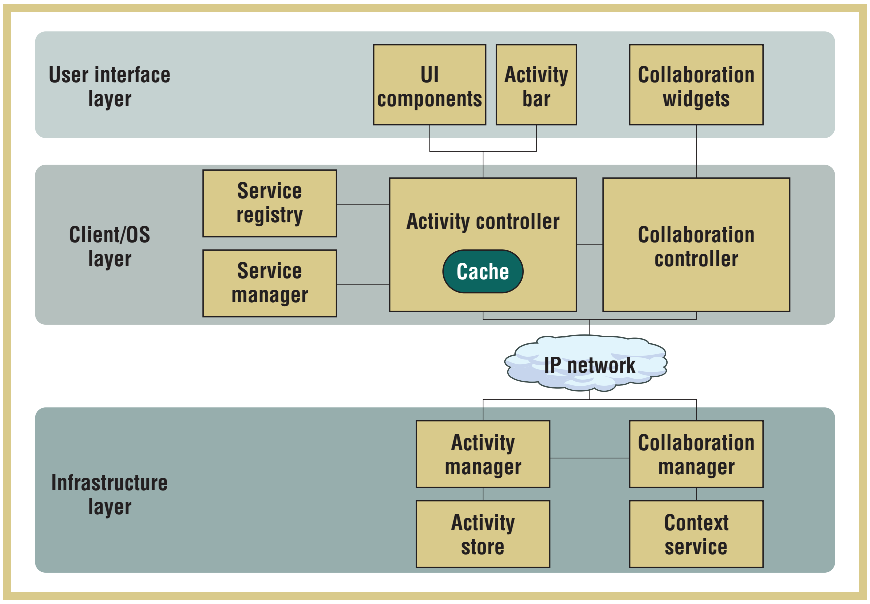


Figure 4. The current activity-based computing architecture.

department and a physician at a ward, to be relevant in the daily working of a hospital.

The use of real-time activity sharing was, furthermore, extended considerably in use. We initially designed and built it for remote collaboration with active participants, such as in the radiology case. During the workshops, however, clinicians used the collaboration support extensively for ad hoc, collocated collaboration. For example, a nurse and a physician both being in the bed ward would have the same activity running on the display in the bed and on a tablet PC. The nurse, for example, would be talking to the patient about medicine while the physician browsed the medical record and made notes.

ABC technology

The ABC framework crystallizes support for the ABC principles we've just described. The framework has a hybrid architecture (see figure 4) using client-server communication for activity and state management and peer-to-peer communication for telepointers, voice link, and other collaborative widgets used in synchronous activity sharing.[5] On the client side, ABC is integrated into the Windows XP operating system and window management system.[3] The client-side architecture has a plugable user-interface layer, which we're using to develop an ABC client application for use in surgical operations. Besides the Windows XP client, ABC clients for the Pocket PC and Symbian mobile phones exist. The infrastructure layer handles activity storage, roaming, and sharing. The activity state is modeled in an XML activity ontology called the Activity Markup Language (AML), which resembles Aura task definition[6] but is more generic in its use. Communication occurs through a dedicated ABC protocol (ABCP) that supports a request-response schema and publish-subscribe notifications. An ABCP handler exists which, for example, makes valid the URL abcp://cfph.au.dk/ms_pedersen. Typing this URL in a browser would activate the ms_pedersen activity. Finally, Java and C# APIs support easy development of ABC-aware applications.

Although we've intensively tested the ABC approach and technology in a laboratory setup, we haven't yet evaluated it inside a hospital. We're currently working on ABC-enabling an EPR and a PACS for deployment in a hospital for further evaluation. In particular, we're focusing on taking ABC into the operating room to test its usefulness in an extreme clinical environment.

To manage complexity in a future where users continuously use numerous computers and devices, we must move computational support from accessing files, applications, and services closer to the users' tasks and goals. We propose ABC as one approach to help users manage this complexity.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for extensive comments that strengthened our focus and presentation. The Danish Center of Information Technology, ISIS Katrinebjerg, and the Danish Research Agency (NABIIT) funded this research. Anders K. Olsen, Jonathan Bunde-Pedersen, and Martin Mogensen have developed parts of the ABC technology, and Claus Bossen has participated in the field work.

REFERENCES

1. H.B. Christensen, "Using Logic Programming to Detect Activities in Pervasive Healthcare," *Proc. 18th Int'l Conf. Logic Programming* (ICLP 2002), LNCS 2401, Springer, 2002, pp. 421–436.
2. D.A. Henderson Jr. and S. Card, "Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface," *ACM Trans. Graphics*, vol. 5, no. 3, 1986, pp. 211–243.
3. J.E. Bardram, J. Bunde-Pedersen, and M. Soegaard, "Support for Activity-Based Computing in a Personal Computing Operating System," *Proc. SIGCHI Conf. Human Factors in Computing Systems* (CHI 06), ACM Press, 2006, pp. 211–220.
4. J.E. Bardram, "Activity-Based Computing: Support for Mobility and Collaboration in Ubiquitous Computing," *Personal and Ubiquitous Computing*, vol. 9, no. 5, 2005, pp. 312–322.
5. J.E. Bardram, J. Bunde-Pedersen, and M. Mogensen, "Differentiating between Accountable and Ephemeral Events in the ABC Hybrid Architecture for Activity-Based Collaboration," *Proc. IEEE Int'l Conf. Collaborative Computing* (Collabo-

rateCom 2005), IEEE Press, 2005, pp. 168–176.

6. D. Garlan et al., “Project Aura: Toward Distraction-Free Pervasive Computing,” *IEEE Pervasive Computing*, vol. 1, no. 2, 2002, pp. 22–31.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.

the AUTHORS

Jakob E. Bardram is a professor at the IT University of Copenhagen. His research interests include pervasive computing, software architecture, computer-supported cooperative work, human-computer interaction, and pervasive healthcare. He received his PhD in computer science from the University of Aarhus. He’s a member of the IEEE and ACM. Contact him at the IT Univ. of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen S; bardram@itu.dk.

Henrik B. Christensen is an associate professor in the Department of Computer Science at the University of Aarhus. His research interests include computer science education research and software architecture with a special interest in methods, tools, and techniques to ensure reliable and flexible architectures. He received his PhD in computer science from the University of Aarhus. Contact him at the Dept. of Computer Science, Univ. of Aarhus, Aabogade 34, 8200 Aarhus N, Denmark; hbc@daimi.au.dk.