# Activity-Based Computing for Medical Work in Hospitals

JAKOB E. BARDRAM
IT University of Copenhagen, Denmark

Studies have revealed that people organize and think of their work in terms of activities that are carried out in pursuit of some overall objective, often in collaboration with others. Nevertheless, modern computer systems are typically single-user oriented, that is, designed to support individual tasks such as word processing while sitting at a desk. This article presents the concept of Activity-Based Computing (ABC), which seeks to create computational support for human activities. The ABC approach has been designed to address activity-based computing support for clinical work in hospitals. In a hospital, the challenges arising from the management of parallel activities and interruptions are amplified because multitasking is now combined with a high degree of mobility, collaboration, and urgency. The article presents the empirical and theoretical background for activity-based computing, its principles, the Java-based implementation of the ABC Framework, and an experimental evaluation together with a group of hospital clinicians. The article contributes to the growing research on support for human activities, mobility, collaboration, and context-aware computing. The ABC Framework presents a unifying perspective on activity-based support for human-computer interaction.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*User interface management systems;Theory and methods;User-centered design*; D.2.11 [**Software Engineering**]: Software Architecture—*Domain-specific architectures*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Computer-supported cooperative work; Collaborative computing; Asynchronous interaction; Synchronous interaction*; J.3 [**Computer Applications**]: Life and Medical Science—*Medical information systems*

General Terms: Human Factors

Additional Key Words and Phrases: Framework, architecture, activity-based computing, cooperation, ubiquitous computing, electronic patient record, activity-awareness

**ACM Reference Format:**

Bardram, J. E. 2009. Activity-based computing for medical work in hospitals. ACM Trans. Comput.-Hum. Interact. 16, 2, Article 10 (June 2009), 36 pages. DOI = 10.1145/1534903.1534907 http://doi.acm.org/10.1145/1534903.1534907

The ABC Project is funded by the Danish Council for Strategic Research.
Author's address: IT University of Copenhagen, Rued Langgards Vej 7, D2300 Copenhagen S., Denmark; email: bardram@itu.dk.

DOI 10.1145/1534903.1534907 http://doi.acm.org/10.1145/1534903.1534907

## 1. INTRODUCTION

An increasing body of research on information workers using personal computers shows that there is a significant mental and manual overhead associated with the handling of parallel work and interruptions [Czerwinski et al. 2004; Mark et al. 2005; Robertson et al. 2004; Adamczyk and Bailey 2004; Iqbal and Horvitz 2007], and that the user interface in current operating systems fails to give adequate support for the resumption of previous activities and for easy alternation between parallel activities [Czerwinski et al. 2004; Robertson et al. 2004].

These systems are typically single-user oriented, that is, designed to support individual tasks such as word processing, handling email, and editing graphics. This personal and task-oriented approach provides little support for the aggregation of resources and tools required in carrying out higher-level activities. It is left to the user to aggregate such resources and tools in meaningful bundles according to the activity at hand, and users often have to reconfigure this aggregation manually when shifting between a set of parallel activities.

A number of studies have revealed that people organize and think of their work in terms of *activities* that are carried out in pursuit of some overall objective, often in collaboration with others [Christensen and Bardram 2002; Gonzalez and Mark 2004; Moran et al. 2005]. Various research initiatives have therefore sought to introduce basic support for "activities" or "tasks" into the computer system. All these initiatives share the overall goal of trying to create computational support that would enable users to handle the complexity of the many different applications, services, documents, files, users, and other materials involved in achieving the objective of a given activity.

The aim of the Activity-Based Computing (ABC) project [Bardram and Christensen 2007] is to investigate how to create computational support for human activities. In contrast to existing approaches, which tend to focus on information workers sitting at a desk, our approach has been to address activity-based computing support for clinical work in hospitals. Once you move away from the desktop and into a non-office-like environment such as a hospital, the challenges arising from the management of parallel activities and interruption are amplified because multitasking is now combined with a high degree of mobility, collaboration, and urgency [Bardram and Bossen 2005]. By "mobility" we mean that clinicians are constantly moving between different physical and social working environments and often between different computational devices as well. Unlike information workers, clinicians in a hospital do not have personal computers. Instead they have to access medical information such as patients' records through shared PCs. "Collaboration" refers to the fact that clinicians need to remain aware of others' work and to be able to coordinate and communicate easily with relevant colleagues. Finally, the "urgency" of clinical work means that the overheads involved in accessing medical information, including manual reconfiguration due to interruptions, mobility, or collaboration, must be kept to an absolute minimum, since delays may have a direct impact on the well-being of a patient.

The approach taken in the ABC project is to design an infrastructure that will enable clinicians to handle a large set of parallel activities spanning multiple services and applications, while moving around inside the hospital and collaborating closely with others. The end-user devices in question are public rather than personal, and include large wall-based interactive displays, touch screens embedded in hospital beds, and smaller mobile devices. Such devices resemble the original proposed Ubiquitous Computing devices of a tab, pad, and wall [Weiser 1991]. In a ubiquitous computing setup of this kind, where users are using a multitude of heterogeneous computing devices, it is essential that they be supported at the overall activity level, for it becomes impossible to use such a ubiquitous computing setup if the user has to rearrange applications and services whenever he or she shifts between computational devices and/or activities. Furthermore, the ubiquitous computing concept of merging the computational devices at hand requires that these devices should be able to adjust themselves to the user according to the particular context and the actual activity that the user is engaged in. Thus, as Abowd and Mynatt [2000] argue, ubiquitous computing systems must strive to support computing at the same level of abstraction as that is involved in human perception, namely at the level of activity.

This article presents *Activity-Based Computing* as an approach to computing that focuses on computational support for handling parallel, mobile, collaborative, and distributed activities, with special reference to the work of clinicians in hospitals. The approach is to represent human activity explicitly as a computational construct and to provide technological support for it. The aim is to create a generic representation of activity that will enable us to integrate the various tools that people use as part of a given activity while at the same time supporting multitasking, mobility, and collaboration.

Five principles underlie the ABC approach: activity-centered resource aggregation, activity suspension and resumption, activity roaming, activity sharing, and activity awareness. All of these principles work together and supplement each other. Broadly speaking, prior research into activity-centered computing has addressed resource aggregation and activity suspension and resumption, but little attention has so far been paid to activity roaming, sharing, and awareness.

These five principles are rather generic in nature, and there is evidence to suggest that activity-based computing support would be beneficial to other application domains besides health care [Bardram et al. 2006; Muller et al. 2004; Moran et al. 2005]. The focus of the present article, however, is on the design, implementation, and evaluation of activity-based computing support for health care work inside hospitals.

This article is organized as follows. First, related work within activity-based computing is presented and we discuss the ways in which our work builds upon and extends these prior efforts. Section 4 discusses the five principles of activity-based computing and reveals their empirical and theoretical roots. Sections 5 and 6 present the ABC framework, which implements support for activity-based computing in a medical domain. This article focuses on presenting the user-interaction design of activity-based computing. Section 7 presents

an experimental evaluation of the ABC framework, and Section 8 discusses the findings from this evaluation. Section 9 concludes the article.

## 2. RELATED WORK

Other systems have been designed to provide more direct support for managing multiple concurrent activities associated with large amounts of digital material. In our discussion below, we categorize these systems as follows: task or activity management systems; inference-based activity systems; virtual desktop management systems; and collaboration support systems.

*Task or activity management systems* record the history of a task, and the virtual workspace associated with it, usually by creating context out of message threads. Email communication threads are stored and reused in TaskMaster [Bellotti et al. 2003] and activity threads comprising of all accessed data objects are stored and maintained in the Activity Explorer prototype [Vogel et al. 2004; Muller et al. 2004]. Recently, the Unified Activity Management (UAM) project [Moran et al. 2005] has proposed a Web-based service for bundling resources such as file attachments and notes into a work activity that can be shared by a set of users much like an extended Team Room system. These systems accurately reflect the history of a project (or task) and the various data objects used in it: contacts, emails, documents, etc. A common feature of such systems, however, is that they are tied to the application that maintains the history and do not support arbitrary use of the computer. The Activity Explorer, for example, supports 6 predefined types of objects: messages, chat, files, folders, screen shots, and to-do items. Hence, users need to "live" [Muller et al. 2004, p. 381] in the application, and information and work tasks that are not handled by the application are not supported. Our approach differs from this in sofar as we seek to enable activity-based computing support at the level of the operating system rather than that of the application [Bardram et al. 2006].

*Inference-based activity systems* monitor and record the user's interaction with the computer in order to draw inferences about the user's activity. The UMEA system [Kaptelinin 2003] records information about a user's low-level operations, such as key strokes and file handling by monitoring the computer's file system, input devices, and running applications. This enables semi-automatic maintenance of the content of project-related pools of documents, URLs, etc. A similar approach is used by the TaskTracer [Dragunov et al. 2005], which is designed to infer so-called task profiles, and by the Context-Aware Activity Display (CAAD) [Rattenbury and Canny 2007] system, which seeks to automate the creation, maintenance, and visualization of tasks. The goal of these systems is to help users to access records of past activities and quickly restore the context of earlier tasks. In a similar way, our principle of activity awareness is designed to recognize relevant activities based on the user's current usage context. But instead of looking at the user's interaction with the computer, our approach looks at the user's interaction with the real world. For example, when the physician or nurse is close to the patient, the computer looks up the patient's medical record. Hence, by adapting the computer system to the

context in which it is used, activity-based computing embraces the conceptual notion of context-aware computing [Dey et al. 2001].

The original Rooms system [Henderson and Card 1986] was the first *virtual desktop management system* to allow users to organize application windows in different "rooms" associated with different tasks. A more recent version of this principle has been presented in the GroupBar system [Smith et al. 2003]. In addition to virtual desktop management systems, a number of research projects have proposed solutions for task management. These include support for moving groups of windows to the desktop periphery in the Scalable Fabric system [Robertson et al. 2004], extending the user's desktop with additional screen space [Baudisch et al. 2001] or peripheral displays [MacIntyre et al. 2001], employing 3D in the TaskGallery window management [Robertson et al. 2000], using time as the main organizing principle in task management [Rekimoto 1999], or allowing for hierarchical window organization and elastic stretching and resizing of windows in the Elastic Windows system [Kandogan and Shneiderman 1997].

Our work differs in at least two ways from the previously described work on new user interface metaphors for task management. First, activities in ABC are *persistent* and *stateful*, which implies that the state of an activity is preserved across service restart or computer shutdown. This is a fundamental difference, because it illustrates that activity-based computing is not designed to support the grouping of application windows in convenient ways, but to support long-lived human activities that unfold and evolve over time. Second, activities are *distributed* across networked computers and thereby support users to move their work activities with them while roaming between devices (i.e., the principle of "activity roaming"). The Gaia architecture for Smart Spaces [Hess et al. 2002] also supports the user in moving applications across different computers, but offers no support for task or activity management in this regard.

With the exception of UAM and Activity Explorer, all these systems are single-user systems and basically extend the single-user operating systems perspective by adding support for handling concurrent tasks and activities. By contrast, the ABC approach seeks to embed *support for collaboration*—activity sharing—as a fundamental aspect of activity-based computing. The techniques used in activity sharing extend from research on application sharing systems in CSCW. More specifically, the ABC framework's support for activity sharing builds on a replicated architecture [Begole et al. 1999] using collaboration-aware applications that are specifically designed for simultaneous use by multiple users. However, in contrast to similar systems such as DistView [Prakash and Shim 1994], GroupKit [Roseman and Greenberg 1996], and Rendezvous [Edwards 1994], ABC extends the support for collaboration from the level of the application to that of the activity. Thus instead of supporting application sharing ABC supports activity sharing, and instead of having collaboration-aware applications, ABC has activity-aware applications. This—we think—significantly extends the work on application sharing in CSCW.

To sum up, activity-based computing builds and expands upon prior work within each of the areas described above: activity management, virtual window management, collaboration support systems, and context-awareness. In

comparison to related work, our approach to activity-based computing supports persistent, stateful, and distributed activities. Moreover, a key contribution of activity-based computing is that all five principles and technologies have been integrated into one another, creating a synthesis that offers a new approach to human-computer interaction. The rest of the article describes this synthesis.

## 3. THEORETICAL AND EMPIRICAL UNDERPINNING

Activity-Based Computing has both empirical and theoretical roots: the former in studies of medical work in hospitals [Bardram 1997, 1998b; Bardram and Bossen 2005; Bossen 2002] and the latter in Activity Theory.[1]

Work in modern hospitals revolves around collaborative problem-solving directed towards the overall objective of treating and caring for patients. From observations and interviews with clinicians it is clear that they always relate to their work in terms of specific work activities, each with a specific objective. The objective often concerns the treatment and care of a patient, but may also relate to other matters such as teaching and research. Moreover, clinicians explain how a series of tasks—such as ordering and analyzing different sorts of examinations—are all part of the treatment of a particular patient. For example, in Figure 1 the nurse and the physician are engaged in a collaborative activity directed towards the treatment of a patient. They are discussing the case and preparing for the ward round, an important action in the overall activity. In preparing for the ward round, they use various artifacts and tools, such as the charts in the medical record laid out on the table, the telephone for communication, and the PC for accessing various digital documents and services. All of these materials, documents, tools, and computer applications have to be used or consulted frequently as part of the treatment of the patient, and hospitals generally expend a good deal of effort on maintaining a collection of relevant materials for use in patient treatment [Bardram and Bossen 2005].

Observations like these accord with Activity Theory, which says that the unit of analysis in understanding human endeavors is an *activity* motivated by the pursuit of a specific *objective*. The activity in question is carried out through a series of *actions* oriented towards specific *goals*, which again are executed through various cognitive and physical *operations* adapted to the concrete *conditions or context* in which these actions are carried out.

It is important to note that Activity Theory does not view an activity as tied to one specific individual. Human activity is always collective and oriented towards an objective that exists outside the individual. Such collaboration is achieved by distributing the actions within a given activity among different actors. For example, the overall treatment of a patient admitted to the ward is divided into actions such as examinations, consultations, prescriptions, medication, blood testing, blood analysis, and care. All of these individual actions are carried out by different clinicians such as the physician, the nurse, the laboratory technician, and the pharmacist. The key challenge in handling this

[1]Activity Theory is a collection of socio-psychological theories primarily originating in the theories of Vygotskij and Leontjev. Kaptelinin and Nardi [2007] provides a good introduction to Activity Theory and its use in interaction design.

Fig. 1. Inside the team room at the ward. Coordination between the physician and the nurse during the ward round in a medical ward.

complex division of labor is to coordinate the individual actions carried out in pursuit of the overall objective of the activity of which they are a part. According to classic Activity Theory communication plays this coordinating role, but if we interpret CSCW research in the light of Activity Theory we see that other mechanisms, such as sharing the object of work, come into play [Bardram 1998a]. The key insight here, however, is that people are able to align their individual actions within an the overall activity only if they are aware of the activity's overall objective and of others' actions in pursuit of it. The classic example in Activity Theory is the battue hunt practiced by the Mongols, which involves gradually encircling the prey through coordinating the movements of a large ring of hunters and "beaters." The beaters involved in the hunt are willing to

scare the prey away, instead of hunting it themselves, only because they are aware of the overall objective of the activity; they know that the hunters are in front of them, and they trust that the spoils, once killed, will be shared. This may sound trivial, but in real-world activities there are numerous examples of individuals who lose touch with the overall objective of their activity and its associated actions. In such cases, their actions may cease to be aligned with the overall activity; they may start to drift or even to work against the overall objective. In a hospital, for example, the care carried out by a nurse and the treatment prescribed by a physician may become misaligned if the nurse tries to help the patient get out of bed while the physician is prescribing intravenous drugs. Hence there is often a need for close coordination between the treatment and care of the patient, which typically happens through communication and sharing of documents, as illustrated in Figure 1.

Following a similar strategy to the UK's NHS Connecting for Health program, Denmark is currently investing in Information and Communication Technology (ICT) to deliver better and safer care for patients in hospitals. An increasing number of clinical computer systems are being introduced in hospitals: electronic medical records (EPR); electronic medication systems (EMS); picture, archiving, and communication systems (PACS); and booking and scheduling systems, to name just a few. Hence, the activities involved in patient treatment are increasingly supported by a wide range of computer systems and applications, each with its own set of patient-related data, clinical functionality and user interface. While these systems indeed provide strong support for various aspects of clinical work, clinicians repeatedly complain about the lack of integration among them: each of them supports an isolated set of work tasks but it remains difficult to align these with the overall activities involved in treating a patient. The main strategy for achieving this kind of integration is to integrate all the underlying servers or databases using a middleware approach [Bernstein et al. 2005]. This principally means trying to achieve integration at the semantic data level—that is, seeking to ensure that the patient's social security number matches throughout the various applications [Ferrara 1998]. Very little effort has so far been directed towards ensuring the integration of the different applications at the level of clinical activities.

The aim of activity-based computing is precisely to help users to manage the complex set of actions, tools, materials, resources, and people involved in an activity by introducing an explicit representation of the activity into the computer system. An important aspect of this approach is that we do not draw any distinction between computerized and nondigital resources and materials: the representation of the activity should encompass. For example, when requesting, taking, and analyzing a blood sample, both the computerized request form and the physical blood sample in the tube are seen as equally important resources in this part of the activity.

In designing activity-based computing support for hospital work, we have employed a user-centered [Beyer and Holtzblatt 1998] and participatory design process [Greenbaum and Kyng 1991; Kyng and Mathiassen 1997]. The ABC framework has gradually emerged over a two-year period and has been designed in close cooperation with a group of 6 clinicians (3 doctors and 3 nurses) working

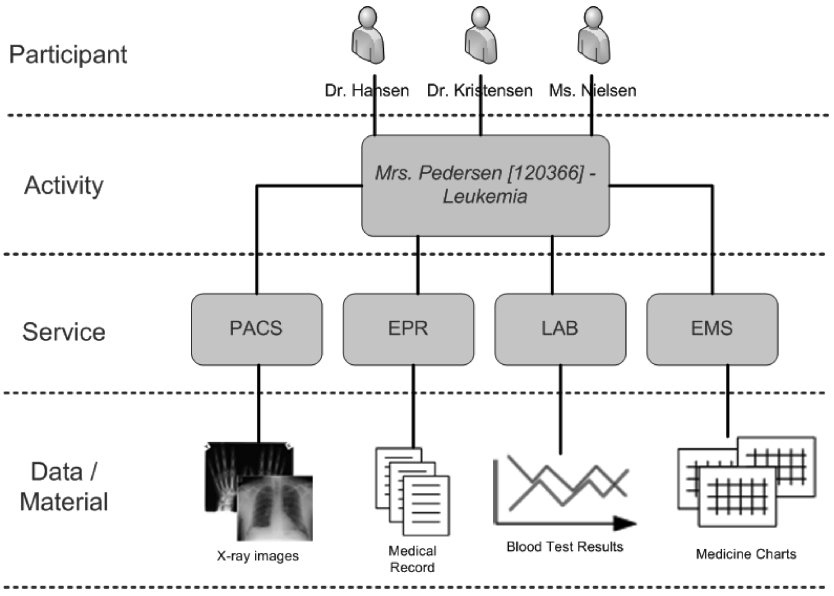


Fig. 2. A computational activity that aggregates a set of computational services, data resources, and users.

at a hematology department in a large teaching hospital. We have conducted 11 design and evaluation workshops in which the clinicians codesigned, used, evaluated, and tested the framework. Each workshop lasted 3–6 hours.

To sum up, the design of the activity-based computing approach has been based on a combination of empirical field studies of medical work in hospitals, theoretical analysis based on Activity Theory, and a user-centered participatory design process involving a group of clinicians over a long period of time.

## 4. ACTIVITY-BASED COMPUTING

In activity-based computing, a "computational activity" (or just "activity") is a computerized representation of a real-world human activity. The purpose of the computational activity is to reflect the human activity and to provide access to resources relevant to its execution:

> *Computational Activity.* An aggregation of services, resources, artifacts, and users that are relevant for a real world human activity.

The computational activity used to represent the work done in Figure 1 is illustrated in Figure 2. This computational activity aggregates and links services, resources, documents, and users that are relevant to the real-world activity of treating Mrs. Pedersen for leukemia. Among other things it gives access to the patient's medical records, information on the medicine administered, and medical images. Access to these materials is mediated by the respective computer systems involved: the electronic patient record system (EPR); the electronic medication system (EMS); and the picture, archiving, and communication

system (PACS). Hence, activity-based computing extends computational support "upwards" from the level of application and document to the level of the overall activity [Bardram 2005a].

In activity-based computing, computational activities become first class entities that are represented explicitly in the computer system. This means that activities are seen as computational entities that can be managed by an infrastructure, an operating system, or some kind of middleware—depending on the manner in which they are implemented. Activities can be persistently saved and distributed via a computer network, and can thus migrate among networked computers. Activities are programmatically accessible through an application programmer's interface (API), which functions just like the API for an operating system. Activities are also directly accessible to users in the user interface. This means that activities can be manipulated directly by users, like files in a file system:

> *Activity-Based Computing.* A computing infrastructure, which supports users to create, save, manage, suspend, resume, move, share, and discover computational activities.

On the basis of our theoretical and empirical background, we have identified five core principles for activity-based computing: (i) activity-centered resource aggregation; (ii) activity suspension and resumption; (iii) activity roaming; (iv) activity sharing; and (v) activity awareness. This section describes these principles in more detail.

### 4.1 Activity-Centered Resource Aggregation

As already discussed, hospitals and clinicians take great care to organize resources, artifacts, and material into appropriate bundles according to key activities. The paper-based medical record is the prime example of this resource aggregation approach, because it embraces all sorts of materials relating to the treatment of a specific patient. The principle of activity-centered resource aggregation simply states that activity-based computing needs to support this bundling of relevant resources and services. In hospitals where a great number of electronic medical systems are used, a given activity might, for example, encompass the electronic patient record (EPR), access to lab request applications, medical images, lab results, medical observations such as blood pressure, etc. These services and resources may come from many different sources but all are part of a single activity, which is what gives them meaning to the clinicians involved. This is illustrated in Figure 2, where the activity aggregates a set of services and resources.

### 4.2 Activity Suspension and Resumption

Clinicians working in hospitals attend to multiple parallel work activities, and often need to switch rapidly from one activity to another. Physicians and nurses, for example, may attend to 10–20 different cases during a single work shift. Because of the collaborative nature of medical work, moreover, they are often interrupted. For example, physicians are often phoned or paged by colleagues

needing to inquire about a certain patient. Using concepts from Activity Theory, in such instances clinicians are involved in more than one activity at a time and responsible for different actions within each of these activities. Activity-based computing seeks to support the management of many parallel activities, each of which is subject to interruption, by enabling an activity to be suspended and resumed later on.[2]

When an activity is suspended, a snapshot of its current state is persistently saved. When the activity is resumed, this state information is used to enable the clinician to continue where s/he left off, giving immediate access to the services and resources s/he was previously consulting. The kind of state, which is relevant to save depends on the type of activity. In the clinical case, state information about each active application and its corresponding data elements is important. This includes state information on the patient and the medical information being used in this activity.

### 4.3 Activity Roaming

Earlier work on supporting human activities or tasks has treated tasks as a cohesive collection of applications on the local computer. When a user refers to a particular task, the system automatically brings up all the applications and files associated with that task. This mechanism relieves the user from finding files and starting applications individually [MacIntyre et al. 2001]. In our work on activity-based computing the core design challenges arise from a continuously changing user context, such as a modern hospital, in which users switch between activities and the computing environments available to them at different points may vary. For example, doing a hospital ward round involves alternating among many different locations, using the devices available at each point. The goal is to enable physicians and nurses first of all to discuss patients in the ward office, using the wall-sized display, the interactive conference table, and their own Tablet PCs; then to move to the patient's bedside, where they will use the built-in computer in the bed; and later on to use public computers in the hallways of the hospital. It should be possible to continue the activities they are engaged in within all these different computational settings.

These examples illustrate a core concept in activity-based computing, namely *Activity Roaming*. This term refers to the migration of activities from one computing environment (e.g., a desktop PC) to another (e.g., the wall-sized display in the team room, or a mobile unit like the Tablet PC). By allowing for both activity roaming and suspension and resumption, activity-based computing enables the clinician to pause an activity on one device and resume it on another, with its previous state fully restored. The system must therefore be able automatically to bring up all the applications and resources associated with the activity, thereby relieving users from manually restoring all the resources and views

[2]Abowd and Mynatt [2000] similarly argue that "interruptions are expected" in ubiquitous computing and, more importantly, that "resumption of an activity does not start at a consistent point, but is related to the state prior to interruption" (p. 43). Furthermore, the support for ubiquitous computing needs to address the fact that "multiple activities operate concurrently" and therefore provide support for "context-shifting among multiple activities" (ibid.).

associated with the on-going activity: in other words, the tools and materials for executing the operations involved in particular actions within the activity are always ready at hand [Winograd and Flores 1993].

### 4.4 Activity Sharing

As discussed in Section 3, collaboration is central to all hospital work, and especially to the treatment of patients. Clinical collaboration ranges from recurrent planned radiology conferences to ad hoc crisis management in acute situations. Indeed, it is difficult to find activities in a hospital that are not collaborative. Hence a core concept in activity-based computing is to support the cooperative nature of real-world human activities. Activity Theory also emphasizes the fact that an activity is object-oriented and has a motive. In the activity-action-operation hierarchy, the motive (why the given activity is being carried out) is associated with the topmost activity level, and a person is able to understand (and perform) an action only if he or she is aware of the overall activity of which it is a part, and thus the reason for doing it. It is therefore important to provide users with the context of the activity they are participating in, in order for them to act appropriately. This becomes even more important when the actions involved in an activity are distributed among several people, because the goal of a given action, and hence the appropriate way to carry it out, can be understood only within the framework of the overall activity [Bardram 1998a]. For example, when the nurse has to administer medicine prescribed by a physician, she needs to be aware of the overall purpose (motive) for giving this medicine, that is, the particular treatment for a given disease.

In activity-based computing, an activity is seen as inherently shared by a set of participants, who can resume and suspend the activity—asynchronously as well as synchronously—and work "inside it." *Asynchronous Activity Sharing* happens when an activity paused by one user is resumed by another. Because the exact state of the activity was recorded when the first user suspended it, the second user will be able to re-establish the activity where his/her predecessor left off. For example, if a nurse resumes an activity that she shares with a physician, she will see any changes that the physician might have made, such as an additional entry in the medical record or a new prescription for medicine.

*Synchronous Activity Sharing* happens when two or more participants in an activity have resumed the activity simultaneously on different client devices. In this case, they are collaborating within the activity and have a synchronized view of what is going on. For example, if the nurse, working in the medicine room, and the physician, working in his or her office, concurrently engage in an activity, they will be able to follow one another's actions. Similarly, if a physician and a radiologist need to discuss X-ray images, they can engage in synchronous activity sharing. An important aspect of synchronous activity sharing is that session management is incorporated into the activity concept, since the activity functions as a collaborative session manager [Edwards 1994; Roseman and Greenberg 1996].

### 4.5 Activity Awareness

In medical work, activities are often tied to a certain physical work context. For example, medicine is usually prescribed when the physician is dealing with a specific patient on the ward, radiology conferences are associated with the conference room, and the distribution of medicines is often tied to physical artifacts such as a medicine tray and the drugs. In activity-based computing, the term "activity awareness" denotes the principle that the computer system maintains information about the users' real-world activities. This awareness is used to give the user easy access to appropriate resources and tools. Links between particular activities and work contexts can be created to facilitate resumption of relevant activities. For example, the activity relating to Mrs. Pedersen can be linked to contexts in which she is physically present. Moreover, tools for suggesting new activities to the user can be created by monitoring the user's context and creating context-aware activity-discovery agents, that is, agents that seek to identify the user's current activity [Christensen 2002; Intille et al. 2003; Koile et al. 2003].

## 5. TECHNOLOGICAL SUPPORT FOR ACTIVITY-BASED COMPUTING

The *ABC Framework* represents our current implementation of the activity-based computing principles described above. The main goal of the ABC framework is to provide a platform for the *development* and *deployment* of computer applications that support the principles of activity-based computing. The framework consists of three parts: (i) a *runtime infrastructure*, which handles the computational complexities of managing distributed and collaborative activities by adapting to the available services or resources in a specific environment; (ii) a *user interface* that supports users in activity management according to the five activity-based computing principles discussed above; and (iii) a *developer's framework (API)*, which allows programmers to create ABC-aware applications that can be deployed in the runtime infrastructure.

Figure 3 illustrates the ABC architecture. It consists of a range of processes running as part of the underlying infrastructure, a set of client-layer processes (i.e., processes at the level of the particular device) responsible for activity management on that device, and a set of ABC-enabled applications with a user interface in the application layer.

The *Activity Store* handles the persistence of activities by providing an interface to create, delete, and get activities. The store keeps track of each user's usage history, enabling the user to move forward and backward through the list of activities. The *Activity Manager* manages the runtime behavior of activities, enabling them to be created, initialized, suspended, resumed, and finalized by clients. The *Collaboration Manager* handles real-time requirements for synchronous collaboration among active participants within an activity. It does this by maintaining a *Session* object for each collaborative activity currently resumed by one or more users at different devices, or by the same user using several devices.

The *Activity Controller* is the link between the client (i.e., the individual device) and the infrastructure. A client's Activity Controller registers itself at

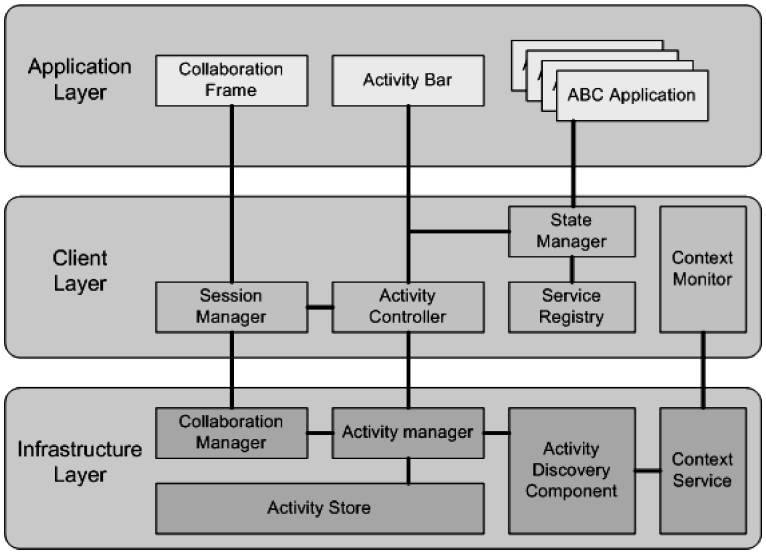


Fig. 3. The ABC Runtime Infrastructure illustrating both server-side and client-side processes.

one or more Activity Managers and maintains a link to the *Activity Bar* (see Figure 5), which is the primary user-interface to the ABC system. The *Service Registry* works as the local service-discovery component on the local client. Thus available services for use in different activities are discovered and registered by the Service Registry. The *State Manager* handles state for the activity. An activity's state is the sum of state information of each of the services participating in the activity, plus certain items of meta-information.

On each device, a *Context Monitor* is responsible for monitoring the context of the device and its use. Context information is collected by the *Context Service*, which cooperates with the *Activity Discovery Component* (ADC). The ADC has a set of rules whereby it is able either to match a set of context events to an existing activity, or to create new activities that seem relevant in a specific context. The context monitor, context service, and ADC support activity awareness.

This article presents version 3.2 of the ABC framework, which is a pure Java implementation. The infrastructure and its processes are implemented in Java; the ABC API consists of Java classes and interfaces; communication between the layers is carried out using Java RMI; and activity and state information is marshaled using Java serialization. The ABC user interface (presented next) is implemented using Java Swing, and all ABC-aware applications are implemented using the ABC Java API. Some of the technical details of the ABC Framework have been described in Bardram 2005a, 2005b; Christensen 2002; Bardram and Christensen 2004.

## 6. THE ABC USER EXPERIENCE

Figure 4 shows the user interface of the ABC client targeted for shared interactive displays within a hospital, such as wall-mounted touch screens, tablet PCs, and bedside displays. As the pictures from our evaluation show (e.g., Figure 9),

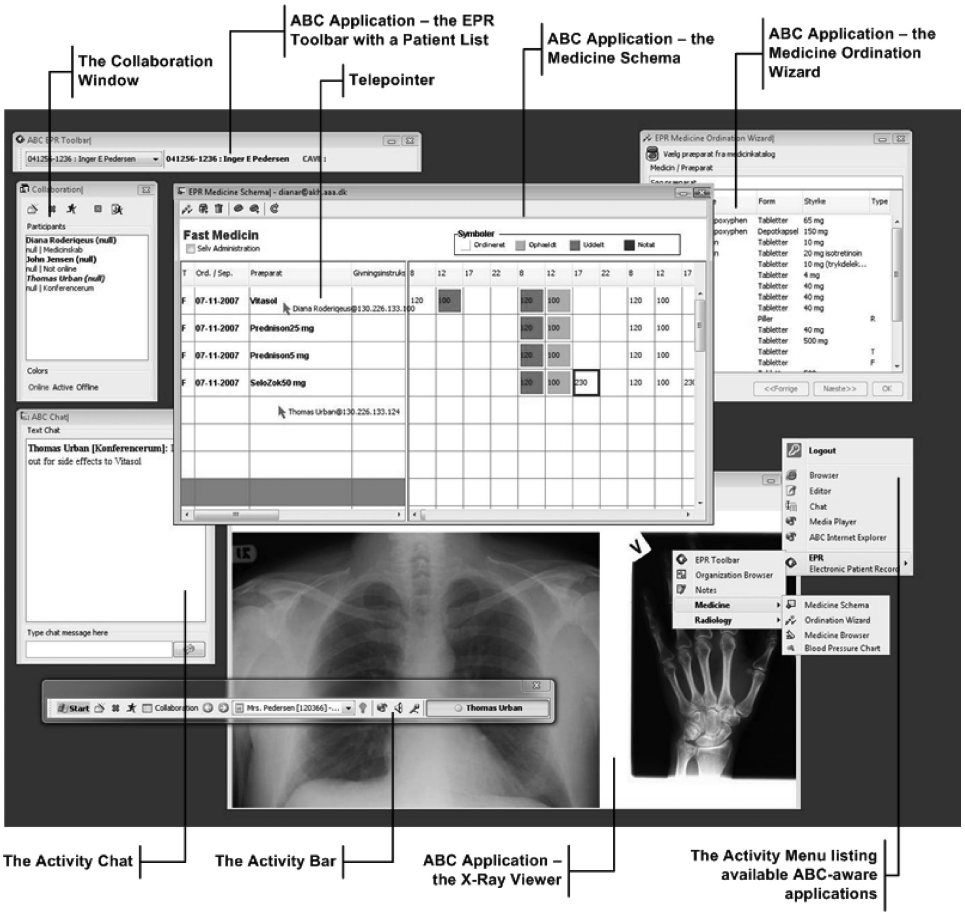


Fig. 4. The ABC User-interface with (i) the Activity Bar (see also Figure 5); (ii) the Collaboration Window showing all participants in this activity; (iii) the Activity Chat for chatting inside an activity; (iv) four ABC Applications which implement the EPR Toolbar, the Medicine Chart, the Medicine Ordination Wizard, and the Radiology Image services; (iv) the Activity Menu; and (v) a Telepointer (see Figure 6 for more details). This picture shows a typical activity in the medical treatment of a patient (Mrs. Pedersen) by three participating clinicians.

the interface takes up the whole screen real estate. Thus the ABC client is not an application but is viewed as the only interface to the device. The widgets from the underlying operating system are hidden, including for example the Windows Taskbar. This is intended to emphasize that the primary entry point to the computer is through activities rather than through applications or files. Figure 4 shows the main UI components of the ABC client: the *Activity Bar* for accessing and managing activities, the *Collaboration Window* showing a list of participants in the activity, the *Activity Chat* used for instant messaging inside an activity, the *Activity Menu* containing a list of all ABC-aware applications available on the given device, and the *Telepointers*, which are used during synchronous activity sharing. In addition, Figure 4 shows a range of

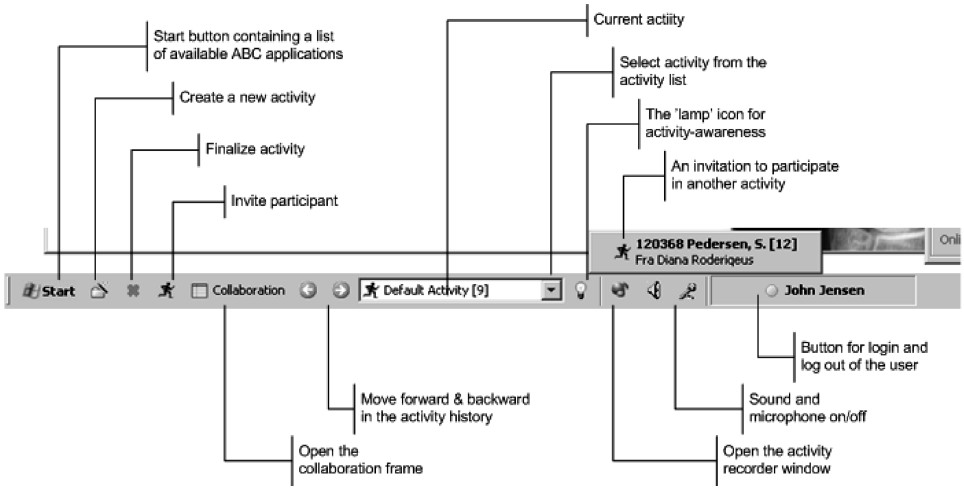


Fig. 5. The Activity Bar, which provides access to the user's list of activities and buttons for activity handling.

*ABC-Aware Applications*, such as the EPR patient toolbar, which lists all patients, the medicine chart, which shows the medicine prescribed for the selected patient, the medicine ordination wizard, and the X-ray viewer for showing radiology images.

### 6.1 Handling Activities

Figure 5 shows the details of the Activity Bar, which is the main interface for managing activities in the ABC Framework. Users interact with the ABC infrastructure through this bar.[3] The current activity is shown in the dropdown box, which can also be used to select another activity. When a new activity is selected, the current one is suspended and stored in the infrastructure, and the new activity is fetched and resumed by the client.

The bar contains buttons for creating and finalizing activities, for inviting new participants to join an activity, and for forward and backward navigation in the Activity History. The microphone and loudspeaker buttons are used to turn sound on and off during synchronous activity sharing. The Lamp icon blinks if the Activity Discovery Component discovers a new activity or if the user is being invited to participate in another activity. When the Lamp button is pressed, these new activities are listed for easy selection. In Figure 5 we can see that the current user (John Jensen) is being invited by Diana Roderiques to participate in an activity regarding the patient Mrs. Pedersen (the activity shown in Figure 4). To the right, the bar shows the name of the person currently

[3]It was a deliberate design decision to have it resemble the Windows Taskbar (or similar bars in different Linux distributions) in order to make it somewhat familiar to users. In a newer version of the ABC Framework, which is closely integrated to Windows XP, the bar has buttons for the most recently used activities, which resemble the buttons on the Windows Taskbar [Bardram et al. 2006].

logged in (John Jensen). By pressing this button, the user suspends his current activity and is logged out of the device.

The Activity Menu can be accessed by clicking on the desktop (as shown in Figure 4) or on the Start button to the left of the activity bar. The Activity Menu contains a list of available ABC-aware applications, that is, applications that understand the ABC API and support activity-based computing, including state and collaboration management. The list reflects the applications available on the particular device, that is, the applications listed in the client-side Service Registry. The user can add an application to the current activity by selecting it from the list, and once started, the application is part of the activity. The resources accessed by an application also become part of the activity. The following scenario illustrates how activity management might take place in a hospital:

> Today, Dr. Urban is responsible for the ward round at Medical Ward A170. On his laptop the activity bar is running and from this he can see a list of ongoing activities that he is participating in. He also notes that he has been invited to a new activity named "120368 Pedersen, S". This relates to a new patient who has been admitted to the ward. He resumes this activity and, before the activity is resumed on the screen, his current activity on another patient is suspended. Using the Start button, he brings up the medical record and the medicine chart, and selects Mrs. Pedersen's file in the EPR Toolbar. After reviewing the patient's status, he logs out of his laptop, leaves it on his desk, and moves toward the ward.

### 6.2 Activity Suspend and Resume

The user can resume an activity by selecting it from the activity list. When an activity is resumed, its state is restored. State information usually includes the look and feel of the user interface; thus when a user resumes an activity the application windows are restored to the same state as when the activity was suspended. In effect, this mimics the virtual desktop metaphor, with each activity corresponding to a virtual desktop with a set of open application windows. Unlike virtual desktop systems, however, ABC also saves state information on the content of each application. In the electronic patient records, for example, state information entails information on what is being looked at, including the patient's ID, current treatment, medication, examinations, request forms, etc. In effect, when an activity is resumed, the content (and not only the look and feel) of each application is restored.

In the medical domain, activity suspension and resumption is designed to help clinicians manage their many parallel activities and the frequent interruptions they encounter. For example, physicians are often interrupted in their current activity by questions from nurses or fellow physicians. Questions are often associated with the treatment of another patient. In these situations, the ABC framework helps the physician to recover the context of each patient simply by selecting the relevant activity from the list of activities in the Activity Bar. And once the interruption has been dealt with, the physician can resume

his or her original activity by selecting it from the list, or by using the back button. In this way, activity suspension and resumption enables users to switch tasks very rapidly in time-critical clinical work.

### 6.3 Activity Roaming

When an activity is suspended, its state is saved in the infrastructure and when the activity is resumed on another device, its state is restored. When a user moves to another device and logs in, the default implementation is to restore the user's most recent activity. Thus the user will find that the activity he or she is currently engaged in "follows" him or her while he or she moves around the hospital:

> In his office, Dr. Urban hits the logout button, thereby suspending the "Mrs. Pedersen" activity. He moves to the A170 ward and enters the ward team room. There he finds the nurse—Ms. Roderiques—who will participate in the ward round that includes Mrs. Pedersen. The doctor and the nurse move to the large wall display and Dr. Urban logs in. This brings up the "Mrs. Pedersen" activity and restores all the applications and the medical data just as they were left in Dr. Urban's office. Dr. Urban adds Ms. Roderiques as a participant to the activity. Together, they discuss the case, log out, and move to Mrs. Pedersen's bedside. Here, Dr. Urban logs in to the bedside display and the "Mrs. Pedersen" activity is restored.

### 6.4 Activity Sharing

In the ABC framework, activities work as a central coordination mechanism for achieving joint activity. Each activity includes a list of participants who have equal access to the activity, all of whom can at any time resume it and work on the various subactions. This is illustrated in the following elaboration of our scenario:

> In the afternoon, the nurse (Ms. Roderiques) resumes the activity in the medicine room, where she prepares the prescribed medicine for Mrs. Pedersen. Later on, when Dr. Urban resumes the activity, he can see that Ms. Roderiques has been engaged in the Mrs. Pedersen activity and prepared the medication prescribed. He invites Dr. Jensen to attend the patient during the evening shift, and leaves a note in the activity chat concerning a potential side effect of the Vitasol prescription.
> In the evening, when Dr. Jensen signs in, he notices that the "lamp" on the activity bar is turned on, and on pressing it sees the invitation to participate in the Mrs. Pedersen activity (as illustrated in Figure 5). Resuming the activity where his colleagues left off, he is provided with a full overview of the status of the treatment, including the ongoing work documented as part of the record, the medical treatment prescribed and given to the patient, and the relevant X-ray images. In the activity chat, he reads the note from Dr. Urban.

This scenario illustrates the case in which participants take turns in working on an activity, which we have labeled *asynchronous activity sharing*. *Synchronous activity sharing* occurs when two or more participants resume the same activity on different devices simultaneously. This may happen both when they are co-located and when they are working remotely. In synchronous activity sharing, the active participants engage in real-time sharing of the activity, which means that status changes are distributed to all participating devices. Thus changes in the window location and size, the content of the applications, the starting and closing of applications, and messaging will all be shared across the devices participating. If participants are not co-located, they can open a voice channel by pressing the microphone and loudspeaker icons on the activity bar. This is illustrated in the following scenario:

> When giving Mrs. Pedersen her medication, Nurse Roderiques observes that it causes vomiting. When she resumes the activity on the wall display in the team room, she notices in the collaboration window that Dr. Urban is online, sitting in his office working on an activity labeled "Hematology Lectures at the University". She decides to talk to him about the problem and invites him to join the activity (even though he is already a participant). On Dr. Urban's activity bar the lamp icon lights up, and when he resumes the Mrs. Pedersen activity, the two users engage in a real-time activity sharing session. They use the voice link, the medicine chart, and the telepointers to discuss the problem and consider how to avoid vomiting. Dr. Urban decides on an alternative drug, and the nurse can see that he is changing the prescription in the medicine chart.

Figure 6 shows how this synchronous activity-sharing scenario looks in the ABC user interface. On the left is the collaboration window which reveals the list of participants and their current status, including active engagement in the activity, current location,[4] and personal status. To the right of Figure 6 we see the medicine chart that Dr. Urban sees on his screen in the scenario above. The window title bar shows that dianar@akh.aaa.dk (Diana Roderiques) is focusing on this window. Focus is revealed by stating user name and by highlighting the window border with a user-specific color. Normally, a window that a given user is focusing on is brought to the front. However, it would be annoying for other people participating in the activity to have windows that other users are focusing on brought to the front of the screen, potentially covering up their own windows. This highlighting approach allows Dr. Urban to use the X-ray viewer without being disturbed by the nurse who is working on the medicine chart, while at the same time being in a position to monitor his colleague's actions.

Figure 6 also shows two telepointers, each with the name of the user and the name of the host machine. We attach the host name because the same user may be using several machines and hence have more than one telepointer.

[4]*Medicinskab* means medicine cabinet, that is, the medicine room, and *Konferencerum* means conference room in Danish.

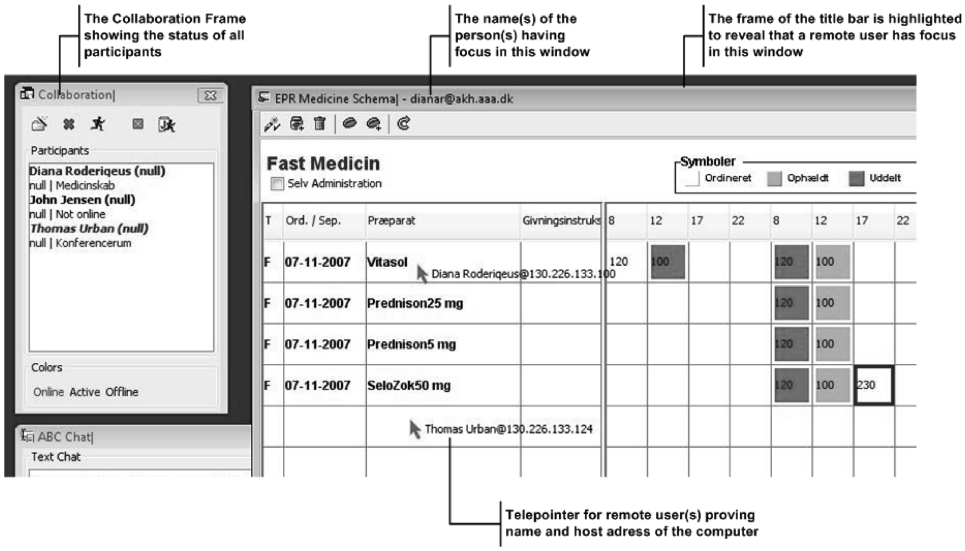


Fig. 6. The collaboration widgets in the ABC framework, including the Collaboration Window, window title bar highlighting, and telepointers.

The activity works as a collaborative session, and the Collaboration Manger handles real-time activity sharing. Since the activity is stored persistently, the collaboration session continues to exist even when there are no active users. By simply resuming an activity, the user joins and participates in a collaborative session, albeit one that currently has only one user. Hence, session management is implicit to the user and avoids the need for explicit session creation, naming, and browsing found in other application sharing systems (e.g., GroupKit [Roseman and Greenberg 1996]). The activity-based session management policy in the ABC framework is a light-weight policy similar to that used in Rendezvous [Edwards 1994], except that in the ABC Framework it works for a whole set of applications, not just one. When a late-coming client joins an ongoing session by resuming the activity, the infrastructure transfers the current shared state of the activity to the new participant without disrupting the work of the existing members, and makes it possible to synchronize state with this new client. State management for collaboration uses the same state management mechanisms that are used for activity roaming [Bardram 2005a].

### 6.5 Activity Awareness

In activity-based computing, *activity-awareness* denotes the principle of adapting the computer system or applications to the activity of the user: for example, finding the relevant medical records during the treatment of a patient. Because activity-based computing has the advantage that the user has already helped define the important activities, the challenge is reduced to creating a linkage between computational and real-world activities.

In the ABC framework, activities can be linked to physical entities by using a unique ID, such as an RFID tag or barcode. This ID is used to associate an

activity with a physical artifact, for example by linking a patient's medicine tray to the activity of handing out medicine to this patient, or by linking a physical X-ray image with an activity representing the radiology examination. When an ABC client reads an RFID tag, for example, it tries to find an activity that matches that tag. Once a match is found, it uses the Lamp icon to signal to the user that a relevant activity has been discovered. The user can then decide to resume the activity if s/he agrees that the computational activity is relevant to his or her current work. For example, when Nurse Roderiques is preparing medication in the medicine room, she will usually use one medicine tray for each patient. Thus when Mrs. Pedersen's medicine tray is placed on the table of the medicine cabinet, the ABC system will prompt her to resume the activity relating to Mrs. Pedersen. Later on, when the same tray is placed close to the touch screen at Mrs. Pedersen's bed, this activity will again be suggested.

Technically, activity awareness through linkage between computational activities and real-world entities is handled by the Activity Discovery Component (ADC) and the Context Service in the infrastructure, which has been implemented using the Java Context-Awareness Framework (JCAF) [Bardram 2005b]. Context events—like the sensing of an RFID tag—are created by the Context Monitor, which sends the event to the Context Service; on the basis of a reasoning engine [Christensen 2002], the ADC then comes up with suggestions for relevant activities. The ADC can either look up activities that are linked to the context event (e.g., an RFID tag), or it can create new activities based on the context information. These activities are handed over to the Activity Manager, which then notifies the relevant clients. The relevance of a client is judged according to where it is located, and who is using it. Thus if the nurse was using a mobile tablet PC instead of the touch screen at the bed, she would still get notification of activities relevant in this context, because the Table PC and the RFID reader in the bed are in the same location.

## 7. EVALUATION

The goal of the ABC framework is to provide a runtime infrastructure, a user interface, and a programming environment that will help create clinical applications better suited to the mobile, collaborative, and hectic work environment of a hospital. In order to evaluate the extent to which the ABC framework meets these goals, we ran a series of experiments aimed at putting to use a set of clinical applications developed using this ABC framework. Since we were especially interested in the benefit to clinicians (rather than programmers) at this stage of our work, we decided to create a suite of clinical applications for hospital use and invited a set of clinicians to evaluate the resulting setup. Thus the focus was on end-user (clinical) evaluation of the runtime infrastructure, the ABC-aware clinical applications, and the ABC user interface, including the activity bar and the collaboration window.

### 7.1 Evaluation Methods

As described in Section 3, the ABC Framework was designed in close cooperation with a set of clinicians from a hematology department. After the period of

codesign and evaluation, this group of clinicians felt confident about the end result (as presented in this article). We then conducted four whole-day evaluation workshops with a group of clinicians who had never seen the ABC Framework before or been introduced to the concept of activity-based computing. This group of 3 physicians and 3 nurses worked in the department of plastic surgery in another hospital. In these workshops, clinicians were asked to role-play a number of clinical scenarios [Bødker and Christiansen 1997] in which they would use the ABC Framework and its clinical applications. By using "situation cards" [Ehn and Sjogren 1991] we introduced several kinds of additional tasks and interruptions into the normal flow of the scenarios. For example, we simulated situations in which the physician is approached by a nurse asking a question, in which he or she receives a telephone call, or in which the nurse is interrupted by some acute patient call. Four researchers participated in these workshops: one running the workshop by initiating the different scenarios and situation cards; one video-recording the workshop; one taking notes; and one acting as a patient. Each workshop ended with a focus group interview. All workshops and interviews were video-recorded and later analyzed by viewing the tapes and taking notes on "important" aspects. Figure 9 and 8 show pictures from the workshops.

### 7.2 ABC-Aware Clinical Applications

In addition to the activity bar, the collaboration window, and the activity chat the following clinically-oriented ABC-aware applications were built to a detail sufficient for supporting the enactment of relevant scenarios. It should be noted that these clinical applications all resemble clinical applications used in hospitals in Denmark.

—*Patient Administrative System (PAS)* contains lists of organizational units, staff, and patients. Includes the EPR Toolbar for easy access to patients.

—*Electronic Patient Medication (EPM) System* includes the medicine chart which provides an overview of the medicine prescribed for a given patient, and tools for the prescription and administration of the medicine.

—*Medical Data Charts* contains charts on a patient's blood pressure, pulse, temperature, weight, etc.

—*Picture Archiving and Communication System (PACS)* provides an overview of a patient's radiology examinations, including images.

—*Electronic Medical Record (EMR)* contains a written record of the patient's ongoing condition and treatment.

The applications are developed as client-server systems that resemble their industrial counterparts. This entails that the ABC framework handles runtime state information on the user interface look-and-feel and the data context of each application, such as the patient and medicine selected. Clinical data, however, are saved in the clinical server. For example, a new prescription made in the medicine chart is saved in the EPM database. In this way, the ABC Framework works independently of the application's architecture and data management strategies.

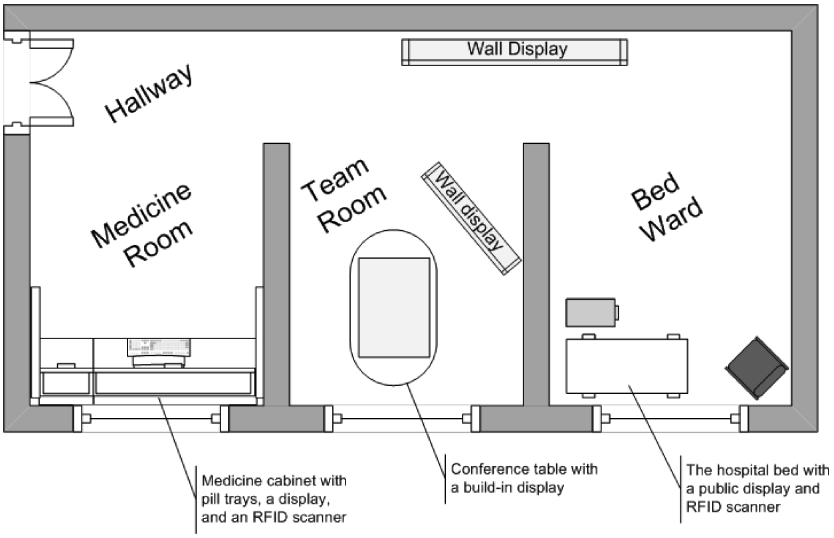


Fig. 7. The layout of the experimental evaluation facility configured to resemble a hospital ward. Contains a medicine room, a team conference room, a room with beds, and a hallway.

### 7.3 Experimental Setup

The evaluation took place in an experimental facility at our university. Due to the nature of medical work, it is not possible to carry out such experiments in a hospital, since this would be too disruptive and invasive of the patients' privacy. Figure 7 shows the basic layout of our test facilities. The layout varied from workshop to workshop, but Figure 7 illustrates the basic components in our test setup. A large room was modeled as a hospital ward containing a room with beds, a medicine room, a small team conference room, and a hallway to connect these rooms. In addition a remote room was used as the physician's office.

### 7.4 Scenarios

The scenarios used in the evaluation workshops were derived from our field studies of the clinicians' daily work and refined in collaboration with them on the day of the workshop. Often the scenarios were based on real clinical cases observed in the hospital. The types of scenarios included:

—*The morning conference*. At the morning conference physicians and nurses discuss the treatment and care of the patients admitted to the ward. This conference takes place in the team room and they use the wall display to discuss the patients and their condition. Each patient has an associated activity which, when resumed, brings up all relevant medical data for this patient.

—*The ward round*. During the ward round the nurse and physician visit all patients at their bedside. Here they use the bedside display to resume the activity relating to each patient, and use the same display during further treatment such as the prescription of medicine. Sometimes this scenario is enacted using a tablet PC.

—*Follow-up on a patient's treatment*. A situation card may be used to pose a question to a nurse or a physician regarding the status of a patient. This was usually handled by resuming the activity in relation to that patient on a vacant public display.

—*Radiology conferences*. A radiology conference typically takes place in the radiology conference room and focuses on the presentation of recent findings in radiology images. Thus radiology images are shown and described by a radiologist while the physicians from the ward listen in. In order to evaluate activity sharing, this scenario was also enacted as a remote conference, that is, with the radiologist sitting in his office while the physicians were gathered in front of the wall display in the team room. Remote conferencing during the evaluation is shown in Figure 8.

—*Medicine administration*. The administration of a prescription for medicine is shown in Figure 9. A nurse resumes activity on the patient in question in the medicine room, prepares the medicine in pill trays, and walks down to the patient's bedside. Here, she again resumes the patient-related activity on the bedside display and uses this to document the patient's medicine intake. This scenario may unfold over a long period of time, may involve different participants, and may be frequently interrupted.

## 8. FINDINGS AND RESULTS

The overall findings from our experiments are that the principles of activity-based computing, and their implementation in the work described, have proved rather robust throughout the design and evaluation workshops. The ABC system unifies computational support for handling multiple applications running on different client machines; it supports mobility through activity roaming between devices; it supports advanced management of parallel work tasks and interruptions; it directly supports both asynchronous and synchronous cooperation, and it makes it possible to link physical artifacts with digital material. In this section we will discuss these findings in more detail.

### 8.1 Activity Management

During the experiments the clinicians were constantly engaged in a wide range of different activities that were used for playing out the different clinical scenarios. These experiments revealed that the clinicians appreciated the notion of gathering patient-related resources and collaborating colleagues into logical bundles (i.e., as single "activities" involving multiple subtasks), and found this system a great improvement on their current working environment. Furthermore, because the ABC framework automatically maintains state information, there was no overhead for the clinicians in handling activities: when a user added a new resource or application (e.g., a medicine chart) to an activity, this was saved as part of the activity's state and later restored once the activity was resumed. All the clinicians reported that this made the "system" (as they referred to it) very easy and straightforward to use.

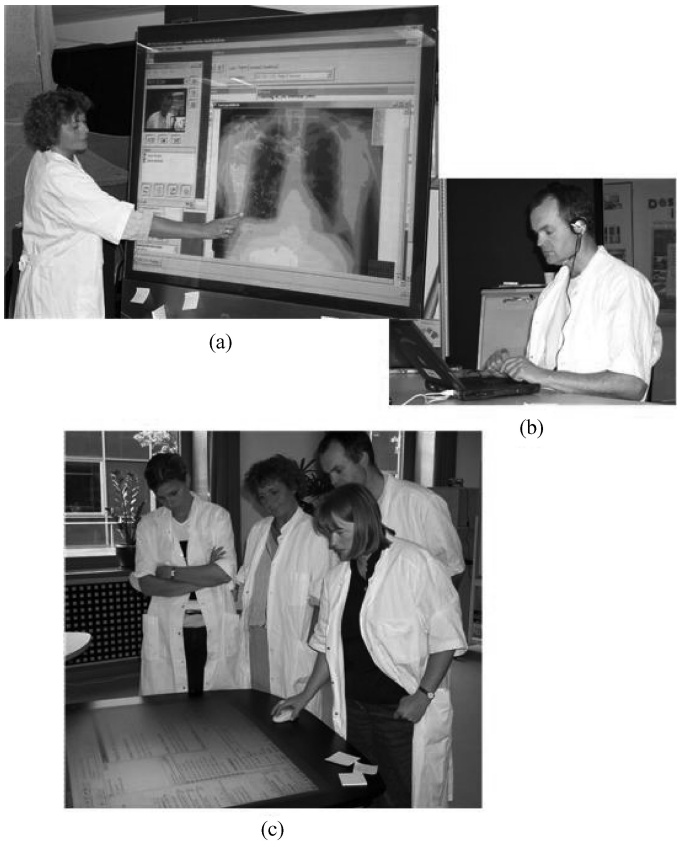

Fig. 8. Evaluation of Activity Sharing scenarios. a), b): A nurse and a physician are engaged in a medical conference using synchronous activity-sharing. The nurse is standing in front of the wall-based display in the Team Room. The physician is in his Office, which is a separate room. c): While still running on the wall-display, the activity is also resumed on the interactive conference table, so that the table and the wall-based display can be used simultaneously.

One conceptual issue was, however, raised. Because an activity carries no semantic information (regarding, e.g., the patient or the treatment), the framework as such provides little support for distinguishing one activity from another. This caused different kinds of problems. One of the recurrent observations was that one activity could "drift" into another—there were no clear demarcation to indicate the ending of one activity, such as the prescription of medicine for a patient, and the start or resumption of another, such as looking at the medical data for another patient. This problem surfaced, for example, when a physician was interrupted by a nurse enquiring about another patient. While working on an activity labeled "Treatment for Mr. Hansen" he might start looking up medical data for Mrs. Pedersen in the medical record. This was a source of much confusion, because the Activity Bar displayed the name Mr. Hansen as part of the activity's name, whereas the EPR Toolbar would show the name Mrs. Pedersen. Furthermore, it was not always easy to judge when to create a new activity—when proceeding to a new patient, but doing the same thing (e.g.,

prescribing medicine for both patients), or when shifting to a new type of task on the same patient (e.g., from the prescription to the handing out of medicine).

In general the clinicians appreciated the flexible support for organizing data and clinical applications into activities and argued that it was an organizational and personal decision as to exactly how the various activities were to be organized and used. But several ways to improve activity management were also proposed and discussed. One of them was to have *Activity Templates* representing frequently used types of activities, such as prescription and handing out medicine. This would enable users to recognize a prototypical real-world activity and easily create a matching computational activity. Another proposal was to replace this pre-hoc denoting of a new activity, such as "now I'm starting on prescribing medicine for Mrs. Pedersen," with a post-hoc bookmark indicating an important point in the unfolding the activity. Thus if the physician has finished taking care of Mr. Hansen, he can bookmark this point, and later on—when he has finished viewing the medical data on Mrs. Pedersen—be able to recollect his treatment of him. A third option would be to use activity discovery: by monitoring user behavior, the activity discovery component could help detect when a user is moving from one activity to another. For example, when the user selects another patient in the EPR Toolbar, the ADC could suggest shifting to one of the activities associated with this patient.

### 8.2 Activity Suspend and Resume

Several of the scenarios involved handling a number of activities in parallel, as happens, for example, when the physician deals with several patients during the ward round. Furthermore, the situation cards introduced several kinds of additional tasks and interruptions into the normal flow of the scenarios. Since clinical work in hospitals is steeped in such interruptions, the clinicians clearly appreciated the mechanism for activity suspension and resumption.

However, the workshops also revealed that the principles of activity suspension and resumption may involve issues of scalability. Thus it is difficult, on the basis of the experimental setups and evaluations carried out in these workshops, to determine how these would scale in real-world deployment. An ABC-aware EPR deployed in a modern hospital would be required to handle a huge amount of patients, users, physical artifacts, and real-world activities, and each user would in no time accumulate a large amount of activities. Hence tools and methods are needed for handling, linking together, and navigating within a complex web of activities. Furthermore, tools for cleaning up—more or less automatically—are essential.

### 8.3 Activity Roaming

The evaluation contained a wide range of scenarios that involved clinicians moving around inside the experimental hospital setup. For example, a ward round scenario would involve the physician and the nurses in constantly moving between the ward, the medicine room, and the team room in order to perform tasks such as prescribing, administering, and handing out medicine. During one workshop we asked the clinicians to use tablet PCs and PDAs for this task. This

workshop clearly revealed that the clinicians preferred not to carry a tablet PC around; it was too heavy it was too large to fit into a white coat pocket, and it was difficult to handle it in front of the patient—the most obvious thing was to drop it in the patient's lap, which was clearly inappropriate. A PDA was considered more suitable, but due to its limited display size, limited battery lifetime, and lack of keyboard, it was perceived as useful only for certain simple tasks, such as checking off medicine intake or doing a simple data entry. Instead, the clinicians preferred using publicly available displays distributed around the hospital for clinical work, and a laptop for more personal work.[5]

Figure 9 shows pictures from the evaluation of activity roaming in which a nurse is handling medicine in the medicine room and another nurse is giving it to the patient. In these mobility scenarios, the ability to suspend work in one location and resume it later on in another was highly appreciated. The combination of activity suspension and resumption and activity-roaming was deemed especially powerful since it allowed one or more collaborating users to get easy and rapid access to the information relating to a particular work activity: the activity would simply be restored in exactly the same state as it was when suspended on another device. Thus when the nurse at a patient's bedside resumed the activity relating to that patient, it would display the medicine chart for the patient that the nurse had used in the medicine room.

The evaluation also revealed certain areas where support for activity roaming in the ABC framework could be improved. One problem arose when activities were roamed between two devices with different display resolution. In the ABC framework, state management handles the look and feel of each application, including the window size and location. This means that if an activity is roamed from a display with a large resolution (e.g., $1900\times1600$) to one with a low resolution (e.g., $800\times600$), a significant portion of the activity's application windows will potentially be left outside the visible area of the display. A related issue arose when clinicians asked for activity roaming between very large devices (e.g., the wall display in the team room) to very small devices (e.g., a PDA or a SmartPhone). In principle, the ABC framework can run on a PDA or a Java-enabled SmartPhone using the J2ME edition. The real challenge, however, is to investigate further what it actually means to roam an activity between two such very different types of devices—especially if we take into consideration that the clinicians saw the small devices as tools for more specific actions within the overall activity. One possible approach may be to support roaming a subpart of an activity to a small device, instead of roaming the whole activity.

### 8.4 Activity Sharing

Figure 8(a)–(b) shows pictures from the evaluation of the conference scenario where a physician in his office and a nurse in the team room are engaged in an online medical conference. This support for distributed, real-time activity

[5]A similar experiment comparing PDAs and built-in displays for medicine hand-out during the ward round has later confirmed that clinicians prefer such public displays to PDAs [Alsos and Svanæs 2006].

Fig. 9. Evaluation of Activity Roaming scenarios. a) : The nurse is standing in the Medicine Room using the computer to prepare the medicine. b) : Another participant in the same activity uses the display attached to the hospital bed to activate the activity and document the patient's medicine intake (picture inserted).

sharing was considered highly relevant in the daily work in a hospital, and in general the native support for cooperation in the ABC framework was highly appreciated as one of its most useful features. During an early phase in our design of the collaborative mechanisms, we used MS NetMeeting for collaboration support. This gave rise to several usability problems in the explicit dial-up establishment of a collaborative session, as well as in the strict floor control mechanism, where only one user can be active at a time. When we introduced replicated collaboration support, using the activity for session management, the distinction between working in a personal or a collaborative mode disappeared. Often, it was transparent to the users whether they were working alone or with others on an activity.

Synchronous activity sharing was initially designed and built for remote collaboration with participants engaging directly in a conversation—as illustrated

in the conference scenario given. During the workshops, however, the clinicians started to use the synchronous activity-sharing mechanisms in other situations. On several occasions, synchronous activity sharing was used to support colocated collaboration in the team room where the same activity was resumed both on the wall displays and on the table display as shown in Figure 8(c). Clinicians at both devices participated actively in the activity, and could mediate their cooperation using the activity-sharing mechanisms. Each user could see what the other(s) was/were doing, for example prescribing medicine using the medicine schema, and could use the telepointers to point to things on the screen. On other occasions, synchronous activity sharing was used for remote social awareness. For example, when the physician resumed an activity at the patient's bedside, he could see that the nurse had resumed the same activity in the medicine room and could follow her handling of the medicine. Thus although they were not engaged in direct collaboration, the activity-sharing mechanisms helped them maintain an awareness of their collaborative work on the activity.

The evaluation also tested the hypothesis that all participants should work on an equal basis within an activity: that is, as long as a person is a participant in a given activity, she or he has access to the whole activity. In this way, the work of the physicians and the nurses is not divided into separate sections, as often happens in medical computer systems. All participants can see and participate in the work of others.[6] This approach was taken in order to help collaborating clinicians to be aware of the overall activity and its objective. Although some of the work done by the physicians may seem irrelevant to the nurse, an awareness of what the physicians are trying to achieve and do is often important in helping her to direct her part of the activity toward the overall objective. For example, the note on side effects from one physician to the other can also be accessed by the nurse (since she is a participant), and it may turn out to be relevant to her work, since she may be the first to notice the side effect in the patient. This hypothesis was confirmed during the experiments. The clinicians argued that it made sense to have a collaborative sharing policy on the activity-management level while retaining accountability within the clinical applications. Accountability was important in relation to patient treatment but not in relation to the way in which clinicians collaborated using the ABC framework.

The evaluation also revealed that the ABC framework failed to give sufficient support for role-based collaboration. Clinical responsibilities and collaboration in a hospital setting are often tied to the different roles that a clinician plays during a work shift. For example, a nurse typically needs to contact and cooperate with the physician on duty, rather than with a specific physician. Thus, drawing on our experience from a hospital setting, we need to extend the user concept to incorporate support for role-based collaboration as well.

[6]Note, however, that the different participants may have different access privileges in the underlying medical applications, such as the medical records. This issue is also handled by the ABC framework but will not be further discussed here. See Bardram [2005c] for more details on this.

### 8.5 Activity Awareness

The concept of linking activities to real-world objects using, for example, RFID tags was also considered very useful. An EPR system contains huge amounts of medical data and navigation around such data is highly cumbersome, especially when you bear in mind that the work context is constantly changing. Hence, inserting physical bookmarks into this huge amount of data was a convenient and easy way to locate the particular medical data relevant to a specific work situation.

Figure 9(a) shows a nurse preparing medicine for a patient by putting pills for the next 24 hours into a medicine tray. These medicine trays are identified by RFID tags. By reading these tags, the medicine cabinet is able to detect the trays for different patients, and display the relevant patient records and his or her medicine chart. Thus when the physical artifact (e.g., the pill tray) is placed close to a public display, the relevant activity is fetched from the infrastructure. In our first prototype, the activity would be displayed "automatically"; that is, there was a direct link between placing a tray in front of the display and resuming the activity.

The evaluation revealed, however, that this design was inappropriate because the nurse would often place several trays in front of, or close to, the display. Thus we discovered that the RFID technology was too coarse-grained and imprecise to be used for automatic activity suspension and resumption. Moreover, it is generally difficult to judge when certain kinds of behavior should trigger an activity shift, and when not [Brown and Randell 2004]. For example, when the nurse places the pill tray for Mrs. Pedersen in front of the display, it may be because she wants to prepare medicine for Mrs. Pedersen, but it may also be because she just placed it there unintentionally. For these reasons, the Lamp icon (see Figure 5) was designed and used to signal activity awareness relating both to collaborating users and to the context.

During the evaluation sessions, this activity-awareness feature using RFID-enabled pill trays was used extensively both in the medicine room and when handing out the medicine to the patient (an RFID reader was also embedded in the bed).

## 9. CONCLUSION

In this article we have presented the principles, design, and evaluation of activity-based computing support for hospital work. The principles of activity-based computing have emerged through ongoing collaboration with clinicians working in large hospitals. Studies and design sessions with these clinicians pointed to a set of challenges associated with the use of computers in hospital work. These include the need for logical integration of digital and physical resources centered around coherent work activities (such as the treatment of a particular patient for a specific disease); the need to support work in a hectic and fast-paced environment with many parallel work tasks and interruptions; the need to support local mobility inside the hospital without burdening the clinicians with heavy computer equipment; the need for remote and colocated collaboration; and the need to integrate physical work involving patients and

medical equipment with the digital representations in the computer systems. Analyzing these challenges to medical work in terms of Activity Theory has helped us identify and specify the five principles of activity-based computing: activity-centered resource aggregation, activity suspension and resumption, activity roaming, activity sharing, and activity awareness.

Related work in activity-based computing has hitherto focused mainly on activity-based resource aggregation (e.g., Unified Activity Management [Moran et al. 2005], Activity Explorer [Vogel et al. 2004; Muller et al. 2004], or TaskGallery [Robertson et al. 2000]) and on activity suspension and resumption (e.g., Rooms [Henderson and Card 1986], or Kimura [MacIntyre et al. 2001]). The work presented here focuses much more strongly on supporting activity roaming, activity sharing, and activity awareness. Furthermore, each of the ABC principles supplements each other and works together. For example, combining the principles of activity-centered resource aggregation and activity suspension and resumption with activity roaming enables clinicians to bundle the resources needed for patient treatment into a single activity, suspend and resume this activity while working on other activities, and take this activity with them while moving around during, for example, the ward round. And by combining the principles of activity-centered resource aggregation and activity sharing, the ABC framework provides native support for computer-mediated collaboration, including session management and state synchronization.

Support for activity-based computing has been implemented in the ABC framework. This article has presented the Java-based version of this framework, focusing especially on the interaction design. Support for activity-based computing is handled through different UI widgets, including the Activity Bar, the Collaboration Window, voice links, and telepointers. The framework also supports linking computational activities to real-world objects; in the current version of the framework this linkage is implemented using RFID technology.

A range of clinical applications were developed using the ABC framework API and were used in an experimental evaluation of the principles and technologies of activity-based computing. This evaluation has provided solid evidence for the applicability and robustness of the activity-based computing principles for medical work in hospitals. In particular the support given by the ABC framework gives in handling a wide range of digital resources, in multi-tasking, and in collaboration was found highly useful. For example, clinicians found it useful to be able logically to bundle together, in one activity, all resources and materials relating to the treatment of a specific patient, including digital resources such as medical records, medicine charts, lab results, and X-ray images. This was especially useful when combined with easy activity suspension and resumption: by a single click in the Activity Bar, the clinicians could now switch work context completely from one patient to another—a task that would otherwise require them to change work context manually in all the different applications.

Support for mobility is central to hospital work. Mobile computing is normally seen as support for portable computers such mobile phones, PDAs, and tablet PCs, which are designed for remote and long-distance mobility. Although the ABC framework does not preclude users from running the Activity Bar and applications on such portable devices, the support for mobility in the ABC

framework is more specifically designed to help users to exploit devices in their vicinity. In the specific hospital case, public displays are distributed and available throughout the hospital. In this sense, the activity-roaming approach is especially targeted towards rich support for local mobility inside the physical and digital boundaries of a large organization like a hospital. As we discovered during our evaluation, the clinicians preferred having this kind of support for mobility to carrying around portable devices such as PDAs, Tablet PCs, and laptops.

Equally important to hospital work is the support for collaboration. But instead of seeing support for collaboration as an external application, the ABC framework integrates support for collaboration as a basic feature. In this way, collaboration becomes the default way of interacting with the computer; as users resume and pause activities, they implicitly engage in a collaborative session involving all the participants in the given activity. The support that the framework gives for activity sharing also turned out to blur the distinction between remote versus colocated and asynchronous versus synchronous collaboration; often we saw activity sharing taking place in all four combinations. This smooth transition between different modes of collaboration turned out to be a strong feature of the ABC framework when compared to more traditional collaboration mechanisms, such as MS NetMeeting. The emphasis on support for collaboration in an ubiquitous computing environment is what differentiates the ABC framework from other ubiquitous computing environments, such as Aura [Garlan et al. 2002; Sousa and Garlan 2002], Gaia [Román et al. 2002], and the Interactive Workspaces project [Johanson et al. 2002].

Compared to office work, hospital work is inherently physical; there are many concrete, tangible "things" involved in patient treatment, including the patient him or herself. Activity-based computing therefore includes support for activity-awareness, meaning that the computational representation of a human activity is somehow aware of its real-world counterpart as, for example, when patient-specific artifacts trigger a display of the computational activity relating to that patient. In this sense, activity-based computing embraces the conceptual notion of context-aware computing [Dey et al. 2001] by adapting an application to the context in which it is used. However, in contrast to the inference-based activity systems discussed in Section 2, activity-based computing has the advantage that the user has already helped define the important activities. This makes activity detection simpler to implement and more accurate in use.

The experiments with the ABC framework also pointed to a range of challenges or areas for improvement. For example, the lack of integration with real clinical applications or other Windows-based applications; the problem of roaming an activity from one display resolution to another; the scalability of many activities; and the lack of support for role-based collaboration. Much of our current work is addressing these issues, and version 4 of the ABC framework now provides native support for the Windows XP operating system, and has different user-interface widgets that improve the user experience [Bardram et al. 2006]. However, the underlying activity-based computing principles and technical components remain intact and unchanged due to the positive evaluations reported in this article.

ACKNOWLEDGMENTS

We acknowledge the important participation of the clinicians from the Hematology Department at the University Hospital of Aarhus who were involved in the project. The ABC concepts have been developed in cooperation with Henrik Bærbak Christensen and Claus Bossen. Henrik B. Christensen, Claus Bossen, Simon Bo Larsen, and Jane Clemensen participated and helped during the evaluation.

REFERENCES

ABOWD, G. D. AND MYNATT, E. D. 2000. Charting past, present, and future research in ubiquitous computing. *ACM Trans. Comput.-Hum. Interact. 7*, 1, 29–58.

ADAMCZYK, P. D. AND BAILEY, B. P. 2004. If not now, when?: The effects of interruption at different moments within task execution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM Press, 271–278.

ALSOS, O. A. AND SVANÆS, D. 2006. Interaction techniques for using handhelds and pcs together in a clinical setting. In *Proceedings of the 4th Nordic Conference on Human-Computer Interaction (NordiCHI '06)*. ACM, New York, 125–134.

BARDRAM, J. E. 1997. Plans as situated action: An activity theory approach to workflow systems. In *Proceedings of the 5th European Conference on Computer Supported Cooperative Work*, T. Rodden, J. Hughes, and K. Schmidtk, Eds. Kluwer Academic Publishers, Lancaster, UK, 17–32.

BARDRAM, J. E. 1998a. Collaboration, coordination, and computer support—An activity theoretical approach to the design of computer supported cooperative work. Ph.D. thesis, Department of Computer Science, Aarhus University, Aarhus. Daimi PB-533.

BARDRAM, J. E. 1998b. Designing for the dynamics of cooperative work activities. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, S. Poltrock and J. Grudin, Eds. ACM Press, 89–98.

BARDRAM, J. E. 2005a. Activity-based computing: Support for mobility and collaboration in ubiquitous computing. *Pers. Ubiq. Comput. 9*, 5, 312–322.

BARDRAM, J. E. 2005b. The Java Context Awareness Framework (JCAF)–A service infrastructure and programming framework for context-aware applications. In *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive '05)*, H. Gellersen, R. Want, and A. Schmidt, Eds. Lecture Notes in Computer Science, vol. 3468. Springer Verlag, 98–115.

BARDRAM, J. E. 2005c. The trouble with login—On usability and computer security in ubiquitous computing. *Pers. Ubiq. Comput. 9*, 6, 357–367.

BARDRAM, J. E. AND BOSSEN, C. 2005. Mobility work—The spatial dimension of collaboration at a hospital. *Comput. Support. Coop. Work. 14*, 2, 131–160.

BARDRAM, J. E., BUNDE-PEDERSEN, J., AND SOEGAARD, M. 2006. Support for activity-based computing in a personal computing operating system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. ACM Press, New York, 211–220.

BARDRAM, J. E. AND CHRISTENSEN, H. B. 2004. Real-time collaboration in activity-based architectures. In *Proceedings of 4th Working IEEE/IFIP Conference on Software Architecture (WICSA '04)*. IEEE Press, 325–329.

BARDRAM, J. E. AND CHRISTENSEN, H. B. 2007. Pervasive computing support for hospitals: An overview of the activity-based computing project. *IEEE Perv. Comput. 6*, 1, 44–51.

BAUDISCH, P., GOOD, N., AND STEWART, P. 2001. Focus plus context screens: combining display technology with visualization techniques. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST '01)*. ACM Press, 31–40.

BEGOLE, J., ROSSON, M. B., AND SHAFFER, C. A. 1999. Flexible collaboration transparency: Supporting worker independence in replicated application-sharing systems. *ACM Trans. Comput.-Hum. Interact. 6*, 2, 95–132.

BELLOTTI, V., DUCHENEAUT, N., HOWARD, M., AND SMITH, I. 2003. Taking email to task: The design and evaluation of a task management centered email tool. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '03)*. ACM Press, 345–352.

Bernstein, K., Bruun-Rasmussen, M., Vingtoft, S., Andersen, S. K., and N¿hr, C. 2005. Modelling and implementing electronic health records in denmark. *Int. J. Med. Inform. 74*, 213–220.

Beyer, H. and Holtzblatt, K. 1998. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann.

Bossen, C. 2002. The parameters of common information spaces: The heterogeneity of cooperative work at a hospital ward. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. ACM Press, 176–185.

Brown, B. and Randell, R. 2004. Building a context sensitive telephone: Some hopes and pitfalls for context sensitive computing. *Comput. Support. Coop. Work 13*, 3-4, 329–345.

Bødker, S. and Christiansen, E. 1997. Scenarios as springboards in design. In *Social Science Research, Technical Systems and Cooperative Work*, G. Bowker, L. Gasser, L. Star, and W. Turner, Eds. NJ: Erlbaum, 217–234.

Christensen, H. B. 2002. Using logic programming to detect activities in pervasive healthcare. In *Proceedings of the International Conference on Logic Programming (ICLP '02)*. Springer Verlag.

Christensen, H. B. and Bardram, J. E. 2002. Supporting human activities—Exploring activity-centered computing. In *Proceedings of Ubicomp 2002: Ubiquitous Computing*, G. Borriello and L. E. Holmquist, Eds. Lecture Notes in Computer Science, vol. 2498. Springer Verlag, 107–116.

Czerwinski, M., Horvitz, E., and Wilhite, S. 2004. A diary study of task switching and interruptions. In *Proceedings of the Conference on Human Factors in Computing Systems*. ACM Press, 175–182.

Dey, A., Abowd, G. D., and Salber, D. 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact. 16*, 97–166.

Dey, A., McCarthy, J., and Schmidt, A., Eds. 2003. *Proceedings of Ubicomp 2003: Ubiquitous Computing*, A. Dey, Ed. Lecture Notes in Computer Science, vol. 2864. Springer Verlag.

Dragunov, A. N., Dietterich, T. G., Johnsrude, K., McLaughlin, M., Li, L., and Herlocker, J. L. 2005. TaskTracer: A desktop environment to support multi-tasking knowledge workers. In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI '05)*. ACM Press, 75–82.

Edwards, K. 1994. Session management for collaborative applications. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, J.B. Smith, Ed. 323–330.

Ehn, P. and Sjogren, D. 1991. From system descriptions to scripts for action. In *Design at Work: Cooperative Design of Computer Systems*, J. Greenbaum and M. Kyng, Eds. Lawrence Erlbaum Associates, Hillsdale, NJ, 241–268.

Ferrara, F. M. 1998. The standard "healthcare information systems architecture" and the dhe middleware. *Int. J. Med. Inform. 52*, 39–51.

Garlan, D., Siewiorek, D. P., Smailagic, A., and Steenkiste, P. 2002. Project Aura: Toward distraction-free pervasive computing. *IEEE Perv. Comput. 1*, 2, 22–31.

Gonzalez, V. M. and Mark, G. 2004. "Constant, constant, multi-tasking craziness": Managing multiple working spheres. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM Press, 113–120.

Greenbaum, J. and Kyng, M., Eds. 1991. *Design at Work—Cooperative Design of Computer Systems.* Lawrence Erlbaum Associates Publishers., Hillsdale, NJ.

Henderson, A. and Card, S. 1986. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Trans. Graph. 5*, 3, 211–243.

Hess, C., Román, M., and Campbell, R. 2002. Building applications for ubiquitous computing environments. In *Proceedings of the International Conference on Pervasive Computing (Pervasive 2002)*. Springer-Verlag, 16–29.

Intille, S. S., Tapia, E. M., Rondoni, J., Beaudin, J., Kukla, C., Agarwal, S., Bao, L., and Larson, K. 2003. Tools for studying behavior and technology in natural settings. In *Proceedings of the International Conference on Ubiquitous Computing*, Ao Dey, Ed. 157–174.

Iqbal, S. T. and Horvitz, E. 2007. Disruption and recovery of computing tasks: Field study, analysis, and directions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, 677–686.

Johanson, B., Fox, A., and Winograd, T. 2002. The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Perv. Comput. 1*, 2 (Apr.), 67–74.

KANDOGAN, E. AND SHNEIDERMAN, B. 1997. Elastic windows: Evaluation of multi-window operations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*. ACM Press, 250–257.

KAPTELININ, V. 2003. UMEA: Translating interaction histories into project contexts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM Press, 353–360.

KAPTELININ, V. AND NARDI, B. 2007. *Acting with Technology: Activity Theory and Interaction Design*. MIT Press.

KOILE, K., TOLLMAR, K., DEMIRDJIAN, D., SHROBE, H., AND DARRELL, T. 2003. Activity zones for context-aware computing. In *Proceedings of the International Conference on Ubiquitous Computing*, A. Dey, Ed. 90–106.

KYNG, M. AND MATHIASSEN, L., EDS. 1997. *Computers and Design in Context*. MIT Press, Boston.

MACINTYRE, B., MYNATT, E. D., VODIA, S., HANSEN, K. M., TULLIO, J., AND M., G. 2001. Support for multitasking and background awareness using tnteractive peripheral displays. In *Proceedings of the ACM Conference on User Interface Software and Technology 2001 (UIST'01)*. 11–14.

MARK, G., GONZALEZ, V. M., AND HARRIS, J. 2005. No task left behind?: Examining the nature of fragmented work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM Press, 321–330.

MORAN, T. P., COZZI, A., AND FARRELL, S. P. 2005. Unified activity management: Supporting people in e-business. *Comm. ACM 48*, 12, 67–70.

MULLER, M. J., GEYER, W., BROWNHOLTZ, B., WILCOX, E., AND MILLEN, D. R. 2004. One-hundred days in an activity-centric collaboration environment based on shared objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM Press, 375–382.

PRAKASH, A. AND SHIM, H. S. 1994. DistView: Support for building efficient collaborative applications using replicated objects. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, J. B. Smith et al., Eds. 153–164.

RATTENBURY, T. AND CANNY, J. 2007. CAAD: An automatic task support system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM Press, 687–696.

REKIMOTO, J. 1999. Time-machine computing: A time-centric approach for the information environment. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology (UIST '99)*. ACM Press, 45–54.

ROBERTSON, G., HORVITZ, E., CZERWINSKI, M., BAUDISCH, P., HUTCHINGS, D. R., MEYERS, B., ROBBINS, D., AND SMITH, G. 2004. Scalable fabric: Flexible task management. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM Press, 85–89.

ROBERTSON, G., VAN DANTZICH, M., ROBBINS, D., CZERWINSKI, M., HINCKLEY, K., RISDEN, K., THIEL, D., AND GOROKHOVSKY, V. 2000. The task gallery: a 3d window manager. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, 494–501.

ROMÁN, M., HESS, C., CERQUEIRA, R., RANGANATHAN, A., CAMPBELL, R. H., AND NAHRSTEDT, K. 2002. A Middleware infrastructure for active spaces. *IEEE Perv. Comput. 1*, 4, 74–83.

ROSEMAN, M. AND GREENBERG, S. 1996. Building real-time groupware with groupkit, a groupware toolkit. *ACM Trans. Comput.-Hum. Interact. 3*, 1, 66–106.

SMITH, G., BAUDISCH, P., ROBERTSON, G. G., CZERWINSKI, M., MEYERS, B., ROBBINS, D., AND ANDREWS, D. 2003. Groupbar: The taskbar evolved. In *Proceedings of the Australian Computer-Human Interaction Special Interest Group (OZCHI)*.

SMITH, J. B., SMITH, F. D., AND MALONE, T. W., EDS. 1994. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. ACM Press.

SOUSA, J. P. AND GARLAN, D. 2002. Aura: An architectural framework for user mobility in ubiquitous computing environments. In *Proceeding of the 3rd Working IEEE/IFIP Conference on Software Architecture*.

VOGEL, J., GEYER, W., CHENG, L.-T., AND MULLER, M. J. 2004. Consistency control for synchronous and asynchronous collaboration based on shared objects and activities. *Comput. Support. Coop. Work 13*, 5-6, 573–602.

WEISER, M. 1991. The Computer for the 21st Century. *Scientif. Ameri. 265*, 3, 66–75.
WINOGRAD, T. AND FLORES, F. 1993. *Understanding Computers and Cognition. A New Foundation for Design.* Addison-Wesley.

Received April 2004; revised December 2007, May 2008; accepted June 2008 by John Stasko