The Activity-Based Computing Project

Jakob E. Bardram
IT University of Copenhagen
DK-2300 Copenhagen, Denmark
bardram@itu.dk

Abstract

This position paper describes the Activity-Based Computing (ABC) project which has been ongoing in Denmark since 2003. Originally, the project took its outset in the design of a pervasive computing platform suited for the mobile, collaborative, and time-critical work of clinicians in a hospital setting. Out of this grew a conceptual framework, a set of six ABC principles, and a programming and runtime framework for the development of activity-based computing infrastructures and applications. Lately, these principles and technologies have been successfully moved to other application areas, and is now used to design and implement activity-based computing support for work in a biology laboratory and for global software development.

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 tasks and interruptions (Czerwinski, Horvitz, & Wilhite 2004; Mark, Gonzalez, & Harris 2005; Robertson *et al.* 2004; Adamczyk & Bailey 2004; Iqbal & 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, Horvitz, & Wilhite 2004; Robertson *et al.* 2004).

Existing operating systems are single-user oriented, i.e. designed to support individual tasks such as word processing, handling email, editing graphics, etc. 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 & Bardram 2002; Gonzalez & Mark 2004; Moran, Cozzi, & Farrell 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 & Christensen 2007) is to investigate how to create computational support for human activities. In contrast to other approaches, which tend to focus on information workers sitting at a desk, our research has taken its outset in the design of 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 multi-tasking is now combined with a high degree of mobility, collaboration, and urgency (Bardram & 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' medical records through shared computers. 'Collaboration' refers to the fact that clinicians need to remain aware of each others' work and to be able to coordinate and communicate easily with relevant colleagues. Finally, the urgency of clinical work means that the overhead involved in accessing and navigating medical information, including manual re-configuration 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, applications, and resources, 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 board, pad, and tab (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, as it becomes impossible to use such a ubiquitous computing setup if the user has to re-arrange applications and services whenever he or she shifts between computational devices and/or activities.

In this position paper, we will describe the conceptual model of activity-based computing, the six ABC principles, and present the current version of our activity-based computing technology.

Activity-Based Computing Principles

On the basis of our theoretical and empirical background, we have identified the *six principles for activity-based computing*. This section briefly describes these principles. The ABC ToCHI paper (Bardram 2009) and the ABC CHI 2006 paper (Bardram, Bunde-Pedersen, & Soegaard 2006) contains a more detailed description of these principles.

Activity-Centered – A 'Computational Activity' collects, in a coherent set, a range of services and data needed to support a user carrying out some kind of (work) activity. For example, in Figure 1 the 'Mrs. Pedersen' activity assembles a set of services and data relevant for the chemotherapy treatment of this leukemia patient.

Activity Suspend and Resume – A user participates in several activities and he or she can alternate between these by suspending one activity and resuming another. Resuming an activity will bring forth all the services and data which are part of the user's activity.

Activity Roaming – An activity is stored in an infrastructure and can be distributed across a network. Hence, an activity can be suspended at one device and resumed on another in a different place.

Activity Adaptation – An activity adapts to the resources available on the device on which it is resumed. Such resources are e.g. the network bandwidth, CPU, or display on a given devices. Hence, an activity might look quite different whether it is resumed on a wall-sized display or on a PDA.

Activity Sharing – An activity is shared among collaborating users. As illustrated in Figure 1, an activity has a group of participants who can access the activity. Consequently, all participants of an activity can resume it and continue the work of another user, thereby taking turns in working on an activity. Furthermore, if two or more users resume the same activity at the same time on different devices, the infrastructure will set up a real-time collaboration session where they all work on the activity at the same time.

Activity-awareness – An activity is 'aware' in the sense that it is able to adapt and adjust itself according to its usage context. Activity-awareness can be used for adapting the user interface according to the user's current work situation by e.g., by showing medical data for the patient currently being treated, or it can be used in a more technical sense, where the execution of an activity, and its discovery of services, is adjusted to the resources available in its proximity.

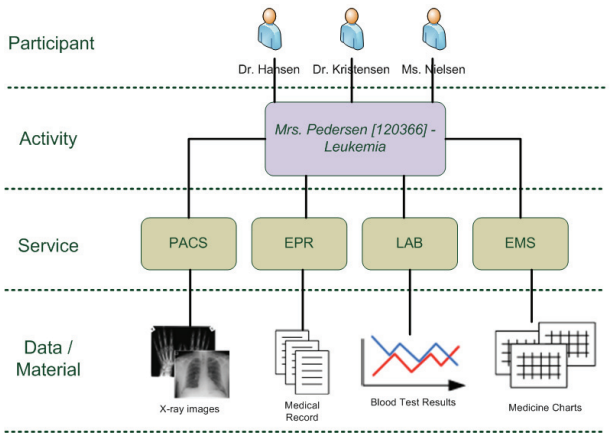

Figure 1: An conceptual illustration of an Activity that has a set of participants, and holds references to a set of services that access a set of resources in terms of medical data items (Bardram 2009).

ABC Ontology

Figure 2 illustrates the current version (5) of the ABC ontology as an UML diagram. The core concept in ABC is the hierarchical structure of `Activity–Action–Operation`. But since these three entities share a lot of common functionality, they share a common abstract superclass called `Enactment`. All enactments (activity, action, operation) share the following:

Lifecycle Methods – Four activity-based lifecycle and one workflow related methods exists:

- **`init()`** – initializes the enactment, making it ready to be used.
- **`finish()`** – finishes the enactment, taking it out of active use.
- **`resume()`** – resumes the enactment.
- **`suspend()`** – suspends the enactment.
- **`done()`** – marks the enactment as done with respect to the workflow status.

Participants – each enactment – and hence each activity, action, and operation – has a set of participants who can access the enactment. Note that a participant is a `Principal`, which can be authenticated to access the ABC infrastructure.

Relationships – As part of supporting activity-based workflow management, the current version 5 of the ABC ontology has been extended with a `Relationship` association. All enactments can have a set of relationships, which again can refer to a set of related enactments. In version 5, a simple association relationship (`AssociationRelationship`) has been mod-

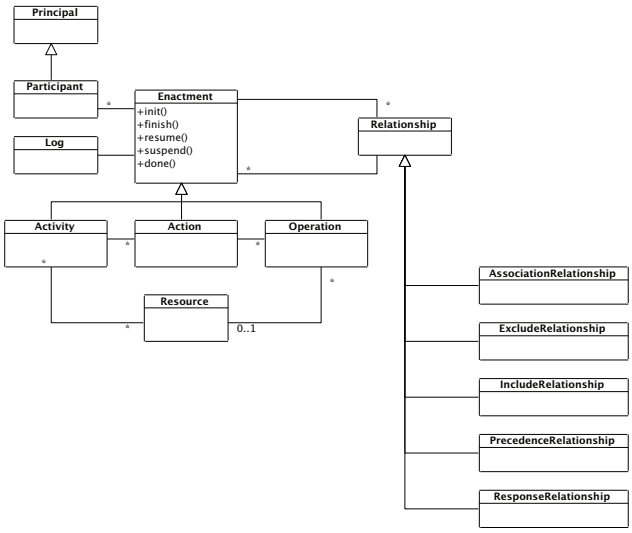

Figure 2: The ABC Ontology – version 5.

elled, as well as the different workflow related relationships, which models whether a set of enactments are either included (`IncludeRelationship`), excluded (`ExcludeRelationship`), has to be temporally preceding this enactment (`PrecedenceRelationship`), or if a response is required from this enactment (`ResponseRelationship`). Activity-based workflow management will not be discussed in any further details here.

Log – A simple log that records what is happening to an enactment. This log is rather generic in nature (and is subject to specialization), and typically logs both technical event (like resume/suspend) as well as human event (like a message associated with the enactment). The log is, for example, used to implement online messaging in activity sharing.

Besides these more general issues handled by the common enactment super class, the `Activity` and `Operation` classes handles the resources:

Activity – An `Activity` has a set of `Resource` objects associated.

Operation – An `Operation` reference at most one `Resource`, which is part of the overall activity. Typically, this resource is 'activated' when an operation is resumed.

Resource – A `Resource` is basically identified and accessible via an Unified Resource Identifier (URI). A resource can be electronic objects like a PDF document, a web page, an email, etc. These kind of resource are typically shown on the display when an operation pointing to this resource is resumed. In version 5, a resource can also be a service, and when resumed, this service is started.

The ABC Infrastructure

Version 5 of the ABC infrastructure is build as a peer-to-peer (P2P) architecture. Figure 3 illustrates the overall P2P architecture. In contrast to previous versions of ABC where the *Activity Manager* was located on a central server, in version 5 each peer (client) has an instance of the Activity Manager running. Each client can also run the user-interface client called the *Activity Browser*. An Activity Browser is per default connected to the local Activity Manager (link 1 in Figure 3). Activity Managers can discover each other (using mDNS), and know the existence of each other (link 2 in Figure 3). An Activity Browser can connect to any Activity Manager which are discovered and enlisted by the local Activity Manager. Hence, in Figure 3, the Activity Browser on client_2 can mount and access data in Activity Manager on client_1 (link 3), once the Activity Manager of client_2 has connected to the Activity Manager of client_1 (link 2).

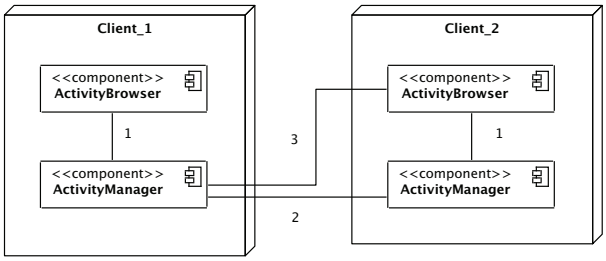

Figure 3: A simple schematic view of the ABC peer-to-peer (P2P) architecture for version 5.

The Activity Manager is build on top of Aexo (), which is an event-based hierarchical data structure enabling the peer-to-peer architecture of the Activity Managers. This peer-to-peer hierarchical mapping structure holds all persistent data in the system including users, activities, (links to) resources, and contextual information such as location.

The Activity Browser is consists of a set of user interfaces designed to be used in large multi-touch displays like wall displays (see Figure 4) and tabletop computers. The implementation is designed for distributed multi-display environment and runs on various hardware devices distributed in a network, e.g. mobile, desktop, table or wall-based display. The system can also be coupled with context sensors such as location trackers and thereby keep track of the physical location of entities like clinicians, patients, equipment, mobile computers, etc.

Activity-Centered

The ABC framework implements the ontology described above, in which activities are the center of the data model. Each activity has a set of actions, which again has a set of operations. Each activity has a collection of *resources*, e.g. files, web pages, applications, medical files record, radiography, etc. The user interaction with the interface starts on the activity level, where activities and related actions are presented in the *Activity View* (Figure 5). In this view, all 'relevant' activities are shown, with their interdependencies

Figure 4: Using the Activity Browser on a large public multi-touch wall display.

and status. Status icons indicates if the activity is resumable and/or done. By clicking the large arrow, a list of actions for an activity is revealed below the activity.

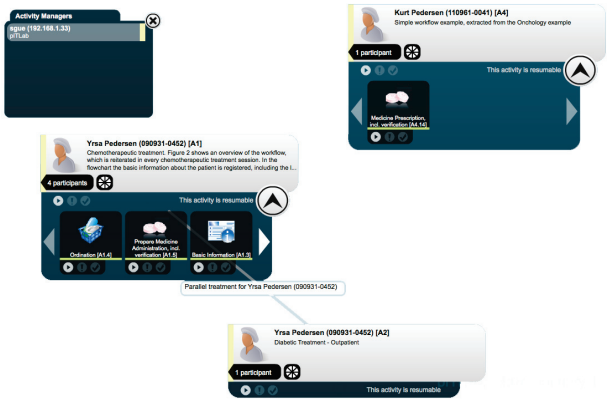

Figure 5: In the *Activity View*, activities are represented by floating panels including graphic and text description. Related actions are linked together.

When clicking on an action, the *Action View* is shown (Figure 6). The action view shows the operations of an action. Each operation is tied to either a resource or a service. As illustrated in Figure 6, an operation can be linked to a Xray image (a resource), in which case the operation shows this image, or it can be linked to a blood pressure monitor (a service), in which case it shown the output from the monitor.

Activity Suspend and Resume

The general ontology and the infrastructure allow for one or more activities to be resumed at the same time. However, the current user interface do not allow the user to do this,

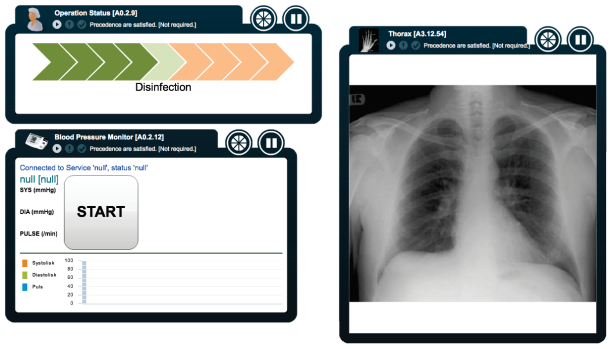

Figure 6: In the *Action View*, an action with its operations are shown. Each operation is linked to a specific resource or service, which is shown inside the window. In this picture, three operations are shown, one showing a resource that reveals the status of the operation; one showing a Xray image; and the last showing the output from a blood pressure monitor service.

since it make little sense to resume the overall activity. Instead, the user can – by clicking the action icon unfolding beneath an activity – resume an action, in which case the display switches to the *Action View* (Figure 6). Each operation shown in the Action View can be individually resumed or suspended. When the user suspends the action, the display switches back to the Activity View.

Activity Roaming

The user can interact with activities he or she is participating in on all activity-based displays. This means that the user can access activities and resume action on any display supporting by the ABC infrastructure, which then allow the

user to roam from one location to another. The infrastructure saves the state of an activity (and its actions and operations), which means that when the user resumed an action, it will appear just as it was left during the previous user session. Note, however, that the previous user session might have been by another participant, in which case the user can pick up where the previous user left the work. Activity roaming is supported by distributing activity data in the peer-to-peer infrastructure of Activity Managers.

Activity Adaptation

Our current approach to adaptation is to design a generic technology that can run on any platforms in the same way. The user interface is designed for interaction on different displays with input specificities, e.g. mouse, touchpad or touch-based input device. This flexibility allows to interact with the activities in several alternative ways. For example, in order to move, suspend or resume the activities or resources on a touch display, the user can either use the mouse buttons or his fingers.

Activity Sharing

The ABC infrastructure has inherent support for collaboration, as activities are shared among participants. All participants of an activity can resume an activity and continue the work of another user or work concurrently. The system also supports activity sharing similar to desktop conferencing systems; users can share the layout of the resources, start multi-site video-conference sessions, and post text messages. The collaborative user interface is shown in Figure 7, which include a video-conference window (top, right), the Action Log for entering text messaging, and a list of participants for this action.

The layout of operation windows is also synchronized between the action views of the participants. This enforces a WISIWYS[1] metaphor for collaboration; whenever a user moves or resizes a operation window, it is moved and resized in other users' action views as well. The framework also allows a user to establish audio and video links with remote participants and hold a multi-point video conference session. The video conference link is established depending on the location of users and the resumed action; the video is automatically started whenever two or more remote users resume the same action, but for co-located participants, video conference is irrelevant and do not start automatically.

Activity Awareness

In activity-based computing, context events, such as the location of users, are linked to the relevance of an activity. For example, the presence of the physician near the patient may trigger a suggestion to resume the computational activity associated with this patient's treatment. In this case, the ABC infrastructure will highlight relevant activities and documents for that patient on any available public display, such as the portable device carried by the physician. The current implementation uses location tracking based on active RFID tags. The user's location is used for several purposes which include updating the list of participants in an activity and starting video-conference sessions between users resuming an activity. Contextual information is stored and managed by a 'Context Service'. The actual detection of relevant activities are done by a specific 'Activity Detection Component' that listens to context events in the context service and finds relevant activities that match with context events. These activities are then highlighted in the Activity View.

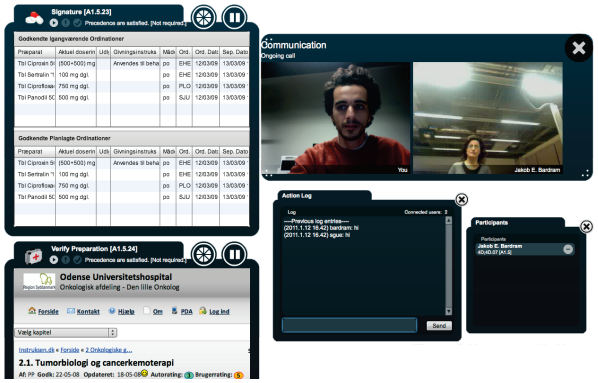

Figure 7: An *Action View* showing operations and their resources on the left, and on the right showing the activity sharing windows; the video-conference window (top), the Action Log for text messaging, and the list of participants.

Ongoing Work

The original research on activity-based computing was done in a hospital environment. Right now we are finalizing the implementation of version 5 of the ABC infrastructure, which then will be put into testing and evaluation in collaboration with a university hospital in Copenhagen. In comparison to other tests of the ABC ideas, concepts, and technologies, we are specifically focusing on designing support for large multi-touch displays and the support for workflow, which has been implemented as part of the TrustCare project[2].

Lately, the ABC concepts and technologies has been used in other projects and application domains as well. As part of the Mini-Grid project[3], an interactive laboratory bench has been build. This bench has a multi-touch table for executing laboratory experiments. The activity-based computing concepts has been implemented as part of this interactive lab bench in order to help organize and manage the different tasks and resources associated with biology experimentation. Right now this lab bench is being deployed in a university laboratory, and we are looking forward to learn about the suitability of using activity-based computing technology for this application domain.

[1] Acronym for "What I See Is What You See", which refers to a paradigm within groupware systems where multiple users, interacting with a multiuser software system, share the same visual perception of the work area at all time.

[2] http://www.itu.dk/research/TrustCare/

[3] http://www.itu.dk/research/mini-grid/

Moreover, as part of the 'Next Generation Technology for Global Software Development' project[4], we are currently investigating the applicability of the ABC concepts and technologies for supporting awareness, coordination, collaboration, and communication in globally distributed software engineering. This research has just started and no result are available yet.

In general, one of the main findings of researching activity-based computing since 2002, is that the concepts and principles of activity-based computing has proved extremely useful and stable across a wide range of challenges and application areas. Moreover, the concepts has helped design new solutions for computational support which has been rather consistent in its logical organization. As such, the concept of 'Activity' and the activity-based computing principles have helped address a wide range of technical problems and has provided a conceptual uniform solution to them.

Acknowledgments

A wide range of people has been part of the ABC project since its start in 2002, and they have all made significant contributions to the research. In order of appearance: Henrik Bæbak Christensen; Claus Bossen; Jonathan Bunde-Pedersen; Mads Søgaard; Afsaneh Doryab; Steffen Sørensen; Morten Esbensen; Søren Nielsen; and Sofiane Gueddana.

The research on ABC has been supported by the Danish Council for Strategic Research as part of the 'Activity-Based Computing' and the 'TrustCare' projects.

References

Adamczyk, P. D., and Bailey, B. P. 2004. If not now, when?: the effects of interruption at different moments within task execution. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, 271–278. ACM Press.

Bardram, J. E., and Bossen, C. 2005. Mobility Work - The Spatial Dimension of Collaboration at a Hospital. *Computer Supported Cooperative Work.* 14(2):131–160.

Bardram, J. E., and Christensen, H. B. 2007. Pervasive computing support for hospitals: An overview of the activity-based computing project. *IEEE Pervasive Computing* 6(1):44–51.

Bardram, J. E.; Bunde-Pedersen, J.; and Soegaard, M. 2006. Support for activity-based computing in a personal computing operating system. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, 211–220. New York, NY, USA: ACM Press.

Bardram, J. E. 2009. Activity-based computing for medical work in hospitals. *ACM Transactions on Computer-Human Interaction* 16(2):1–36.

Christensen, H. B., and Bardram, J. E. 2002. Supporting Human Activities - Exploring Activity-Centered Computing. In Borriello, G., and Holmquist, L. E., eds., *Proceedings of Ubicomp 2002: Ubiquitous Computing*, volume 2498 of *Lecture Notes in Computer Science*, 107–116. Gothenborg, Sweden: Springer Verlag.

Czerwinski, M.; Horvitz, E.; and Wilhite, S. 2004. A diary study of task switching and interruptions. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, 175–182. ACM Press.

Gonzalez, V. M., and Mark, G. 2004. "constant, constant, multi-tasking craziness": managing multiple working spheres. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, 113–120. ACM Press.

Iqbal, S. T., and Horvitz, E. 2007. Disruption and recovery of computing tasks: field study, analysis, and directions. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, 677–686. New York, NY, USA: ACM.

Mark, G.; Gonzalez, V. M.; and Harris, J. 2005. No task left behind?: examining the nature of fragmented work. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, 321–330. ACM Press.

Moran, T. P.; Cozzi, A.; and Farrell, S. P. 2005. Unified activity management: supporting people in e-business. *Commun. ACM* 48(12):67–70.

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 *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, 85–89. ACM Press.

Weiser, M. 1991. The Computer for the 21st Century. *Scientific American* 265(3):66–75.

[4] http://global-interaction.org/en/Research/GIRI-Research-Projects/Global%20Software%20Development