Activity-Based Collaboration for Interactive Spaces

Jakob E Bardram, Morten E Esbensen, Aurélien Tabard

Activity-Based Collaboration for Interactive Spaces

Jakob E. Bardram, Morten Esbensen, and Aurélien Tabard

Abstract Activity-based computing (ABC) is a conceptual and technological framework for designing interactive systems that offers a better mapping between the activities people conduct and the digital entities they use. In ABC, rather than interacting directly with lower-level technical entities like files, folder, documents, etc., users are able to interact with 'activities' which encapsulate files and other low-level resources. In ABC an 'activity' can be shared between collaborating users and can be accessed on different devices. As such, ABC is a framework that suits the requirements of designing interactive spaces. This chapter provides an overview of ABC with a special focus on its support for collaboration ('Activity Sharing') and multiple devices ('Activity Roaming'). These ABC concepts are illustrated as implemented in two different interactive spaces technologies; ReticularSpaces [3] and the eLabBench [21, 22]. The chapter discusses the benefits of activity-based collaboration support for these interactive spaces, while also discussing limitations and challenges to be addressed in further research.

Jakob E. Bardram
Department of Applied Mathematics and Computer Science, Technical University of Denmark
e-mail: jakba@dtu.dk

Morten Esbensen
Pervasive Interaction Laboratory, IT University of Copenhagen, Denmark
e-mail: mortenq@itu.dk

Aurélien Tabard
Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France
e-mail: aurelien.tabard@liris.cnrs.fr

Contents

1 Introduction

Since the pioneering research on ubiquitous computing environments done at Xerox PARC in the early 1990s [23], there has been a growing scientific and commercial interest in Interactive Spaces. In contrast to a single interactive device – such as laptops, tablet computers, interactive displays, smartphones, or tabletop displays – an interactive space is *comprised of several collaborating interactive devices* of many form factors that work together to form a unified and shared interactive experience for several users. Fig. 1 shows a sketch of an interactive space in which several users inside and outside a room can work together by interacting across several devices with different form factors, including wall-based displays (low and high resolution), tabletop displays, and portable laptops and tablet computers.

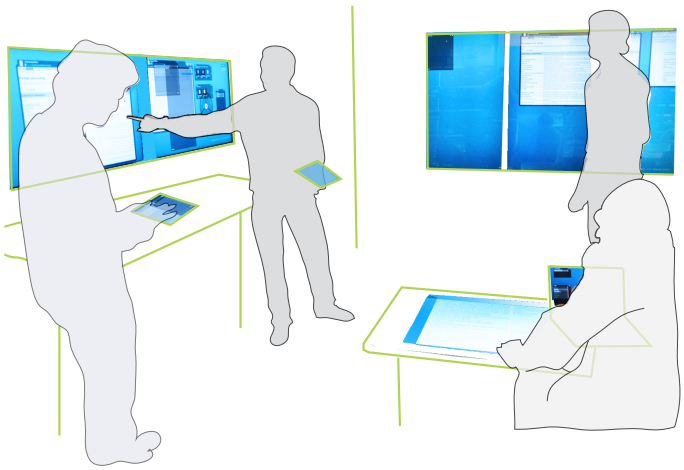


Fig. 1 An *Interactive Space* is comprised of a set of interactive wall, tabletop, desktop, laptop, and tablet that work together in unison. Nomadic users can bring devices to the space, which are then included into the interactive space setup. Collaboration across two or more interactive spaces in different locations can be initiated.

An early example of systems support for interactive spaces is the *i-LAND* system [20]. i-LAND supported interactive surfaces integrated in the architectural space and furniture of a 'smart room', including walls, tables, and chairs. The system allowed users to transfer documents and windows between different surfaces, as well as replicating documents and windows across several surfaces. This allowed users to interact simultaneously on multiple displays: users can make remote annotations at the wall display from one of the interactive chairs or manipulate an artifact at the table. As such, the i-LAND system provided a *unified* interface enabling collocated and synchronous group collaboration. Similarly, *iRos* was a suite of systems components that was designed to help create applications for multiple devices with the ability to integrate portable devices in an interactive space [15]. It supported redirection of input via the *PointRight* component [16]; replication of content with

the *Multibrowse* component [15]; and asynchronous exchange of documents with the *DataHeap* component. In iRos, information could be accessed across multiple displays and by mobile devices dynamically added or removed as they join or leave the interactive space.

Interactive space technologies have also been designed for specific application areas. For example, the *Impromptu* system was designed specifically to support collocated collaborative interaction in software engineering [9]. Software developers could exchange application windows by replicating them, e.g. for problem solving, or by sharing them on public displays, e.g. for discussion or reflection. To improve a collaborative interactive space experiences, Impromptu integrated special collaboratives tools, such as tele-pointers, screen sharing and instant messaging, into the interactive room technology. Interactive space technology have also been designed for clinical work in hospitals. For example, Clinical Surfaces [6] allowed clinicians to manage, access, and move patient data across a distributed multi-display environment covering an entire hospital. The system aggregated medical information relevant for a patient case and allowed clinicians to easy access this patient data across large wall-based displays situated in e.g. the patient wards, the nurses' offices, and inside operating rooms.

The promise of interactive spaces is that users will be able to fluently and flexibly utilize many different interactive devices inside a room according to their need and work activity. By being able to transfer work and use any device – both fixed and portable devices – in an interactive space, users become independent of the limitations of devices and should be able to work more efficiently together and share work. However, with the introduction of multiple devices with different form factors, used by multiple users, for multiple activities, the concept of interactive spaces introduces a new level of complexity both at the architecture level and the interaction level. It is by no means a trivial task to design the advanced interaction techniques needed for sharing, moving, and interacting with files, folders, documents, etc. across multiple devices, users, and locations as part of a collaborative work activity.

To address this challenge of complexity, our group has been researching the concept of *Activity-based Computing* (ABC) [5, 7] and how ABC can be applied in the design of distributed user interfaces for interactive spaces [2, 3, 4, 14]. The core idea in ABC is to explicitly represent the human, collaborative activity as a first class object in the computer and in the user interfaces on the interactive devices making up the interactive space. Hence, just like the desktop windowing systems of a personal operating system uses folders and files to organize computational objects, an activity-based computing system uses '*activities*' and '*resources*' for this purpose. The main idea is then, that the computational activity reflects the real-world activity, which a group of users are involved in.

In this chapter, we will outline the principles of ABC and show how this has been applied to the design of interactive space technology. In particular, we shall focus on how ABC provides support for moving digital resources across multiple devices inside and between interactive spaces – which is called '*activity roaming*' in ABC – as well as how ABC provides support for collaboration between multiple users inside and outside an interactive space – which is called '*activity sharing*' in ABC. We

illustrate the use of these ABC principles in the design and implementation of two specific interactive spaces; the *ReticularSpaces* system which is a general-purpose interactive space technology similar to e.g., i-LAND and iRos, and the *eLabBench* which is a application-specific interactive space for biology experiments in a wet laboratory.

2 Activity-Based Computing

Early research in the 1980s pointed out that activities performed by users of computer systems show complex patterns of interleaved activities, and that contemporary human-computer interfaces provided little support for the kinds of problems users encounter when attempting to accomplish several different tasks [1, 18]. Since then, a large number of observational studies have shown that users often try to structure their work within the context of higher-level activities in desktop environments [8, 10, 12]. Users not only reason within the context of activities, but sometimes also actively tweak available tools to organize their files and folders according to these activities.

These observations have lead to a research agenda on *Activity-based Computing* (ABC) that seeks to make *activities* first-class computational objects. One of the earliest implementations of this idea was the Rooms system [13]. For a historical overview of recent ABC systems, see Bardram et al. [7]. Our group has been researching ABC for more than 10 years with a special focus on providing ABC support for ubiquitous computing [11]. The central goal is to provide a computing platform which allows the user to focus on higher-level collaborative activities rather than low-level application and data management.

The core idea of ABC is the principle of *Activity-centric Resource Aggregation*. This principle states that all documents, files, resources, services, etc. that are relevant for a human activity, should be organized into a corresponding *Computational Activity*, or just *Activity*. Each activity contains a set of participants that are associated with that activity as part of a their collaborative work, and each participant can *resume* and *suspend* the activity as part of this work. An activity can be suspended on one device and resumed on another device, and hence activities *roam* between devices in an interactive space and between different spaces. When resumed on heterogeneous devices in different work context, an activity needs to be able to *adapt* to different computational settings in which it is being resumed and used. Finally, multiple activities are *interlinked* and specific relationships may exist between them. For example in a workflow system, one activity needs to precede another activity.

ABC can be implemented in a variety of ways. Fig. 2 illustrates a minimalist approach to an ABC framework used in the context of a biology laboratory. Here activities are mostly used as a means to aggregate resources and roam them across different devices. In the section below we outline the general principles of ABC and in Section 3 we present how these principles were incorporated in the two different interactive spaces ReticularSpaces and eLabBench.

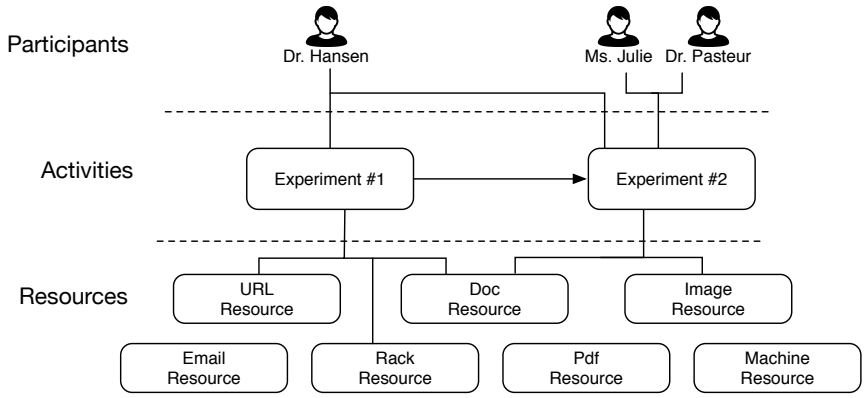
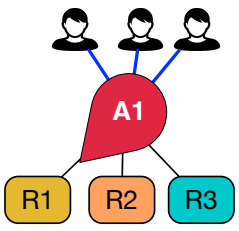


Fig. 2 A *Computational Activity* encapsulates *resources* and *participants* (users) in one coherent first-class object. This ABC model illustrates biology experiments as used in the eLabBench project. The model contains two activities (Experiment #1 and #2), which each aggregates a set of resources such as web pages (URLs), emails, and documents (such as PDF and Word files). Resources can be shared between activities (such as the *doc Resource*). Each activity has a set of participants: Experiment #1 has one participant (Dr. Hansen), whereas Experiment #2 has all three users as participants. Finally, the arrow between Experiment #1 and #2 illustrates a simple workflow relationship meaning that experiment #1 has to be done before #2.

2.1 ABC Principles

The core ABC concepts as outlined above have been crystallized into five ABC principles [5]. These principles are grounded both in theoretical models of human cognition and activity, as well as in empirical research involving the design and evaluation of ABC technologies and applications. Although different types ABC technology have evolved over time, these principles and the core concepts of ABC have remained stable.

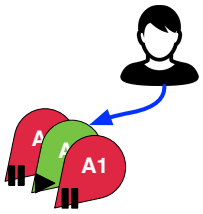
Activity-centric Resource Aggregation



As already discussed, users organize files, folders, and other digital resources into appropriate bundles according to key activities. These resources may come from many different sources and applications (such as files, email, project management tools, editors, etc.) but all are part of an activity, which is what gives them meaning to the user. In ABC all digital resources are organized into *activities*, which are higher-level computational constructs that encapsulate all resources, tools, services, and communication mechanisms into one goal-oriented interaction model. By moving away from classic application-oriented interfaces users are presented with logical units of work rather than the tools required to perform that work. This is especially relevant in

multi-device interactive workspaces in which tools will change depending on the platform.

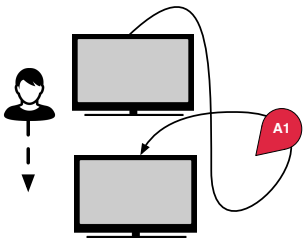
Activity Suspension and Resumption



In all modern work settings, users attend to multiple parallel work activities and often need to switch seamlessly from one activity to another. The biologist, for example, may switch between different lab experiments, teaching classes, and administrative work during a work day. Since users are involved in several collaborative activities, users often interrupt each other. ABC 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. 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 user to continue where s/he left off, giving immediate access to the services and resources s/he was previously using. The specific application domain and the nature of the activities and workflow determine what state information that is relevant to save when suspending an activity. In the biology case, state information about the current experiment and what resources are being used is saved, since this allow a biologist to resume an experiment later and access all resources. In practice this means that on the eLabBench, when resuming an experiment all resources (like pdf documents, spreadsheets, documents, and web pages) are shown in the exact same location and showing the same data, as when suspended earlier.

By supporting activity suspension and resumption, users can easily switch between different activity contexts. Suspending an activity means its state is stored and removed from the active workspace, while resuming an activity restores it. This feature supports parallel activities (multitasking) and interruptions in work.

Activity Roaming



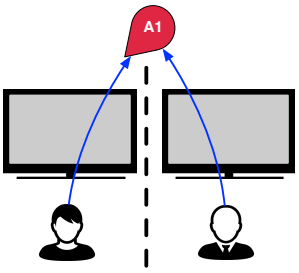
Most of earlier work on activity-centric computing have modeled a task as a collection of applications on the local computer [7]. For example, when a user refers to a particular task, the Kimura system automatically brings up all the applications and files associated with that task. This mechanism relieves the user from finding files and starting applications individually [17]. In interactive spaces, a core design challenges arise from the need to support nomadic and mobile computing in which users' context is changing as users move between different interactive devices.

For example, in biology work, a post doc researcher would conduct the same activity in a variety of contexts: working in his office to plan an experiment, checking material in the wet laboratory, in meetings to discuss the experiment, and finally tutoring students on how to conduct it. While roaming these different physical

places s/he will access different devices including desktops, laptops, interactive lab benches (such as the eLabBench), and wall-based displays. It should be possible to continue the activities that s/he is engaged in within all these different physical and 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 classroom).

Moreover, by *combining* activity roaming with activity suspension/resumption, ABC enables a user to pause an activity on one device and resume it on another, with its previous state fully restored. The system will automatically bring up all the resources and services associated with the activity, thereby relieving users from manually restoring all the resources and views associated with the ongoing activity: in other words, the tools and materials for executing the operations involved in particular actions within the activity are always ready at hand [24].

Activity Sharing



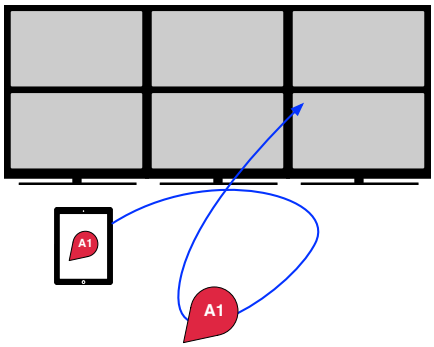
Collaboration is central to all work; yet task- or activity-centric computing approaches are mainly targeted personal information management [7]. In ABC, each activity is *shared* amongst a set of participants as illustrated in Fig. 2. This means that each participant can resume and access the activity. Depending on the timing of each participant's resumption, two types of activity sharing can take place; asynchronously and synchronously.

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 the biology student resumes an activity that she shares with the post doc, she will see all the material and resources that the post doc might have prepared for her.

Synchronous activity sharing happens when two or more participants work on the same activity simultaneously. This can happen collocated on the same device, such as when two biologist work in front on an eLabBench; it can happen collocated on different devices, such as shown in Fig. 3 where multiple users are engaged in the same activity on multiple devices inside an interactive space; and it can happen when users are not collocated, but working remotely from two different devices, such as when a participant outside the interactive space in Fig. 3 accesses the activity unfolding inside the space. In the case where the same activity is resumed on multiple devices — both collocated and remotely — participants are collaborating within the activity and will see a synchronized view of what is going on. While a student is working on an experiment in the lab, a post-doc can add some explanatory resources from his office. In the other direction, the post-doc or professor can follow notes and pictures captured by the student from the eLabBench, as they are produced. An important aspect of synchronous activity sharing is that collaboration

session management [19] is incorporated into the activity concept, since the activity functions as a collaborative session manager.

Activity Adaptation



The principle of activity adaptation supports *multi-device configuration*. When an activity is resumed in an interactive space, the different resources and service may be resumed on different devices, which then is synchronized by the overall activity. For example, in the *ActivitySpace* system, an activity resumed on a tabletop would allow users to include portable devices like smartphones and tablet computers as auxiliary displays showing some of the resources (e.g. images) in the activity [14].

An interactive space is comprised of many different types of devices with many different form factors and capabilities in terms of hardware resources, connectivity, screen size and resolution, interaction modalities (e.g. touch-based), and operating systems. In order for activities to be able to roam between devices they need to adapt to the capabilities of these devices. This can be *technical adaptation* to the resources, connectivity, and sensors available on a device, as well as *user interface adaptation* to the different interaction and displays capabilities of an interactive device inside a interactive space. Hence, an activity might have different (technical) resources available and may look quite different, depending on whether it is resumed on a wall-sized display, a tabletop, or on a smartphone.

3 Interactive Space Technology Cases

The ABC principles have been applied in the design of several systems since they were originally outlined in 2002. In this chapter we shall present and discuss two cases where ABC have been applied in the design of Interactive Space technology: the *ReticularSpaces* and the *eLabBench* systems. These two systems have been documented in detail elsewhere and in this chapter we shall only discuss the role of ABC in their design. The two systems are quite different in how they use ABC; ReticularSpaces is a general purpose platform and user interface for interactive spaces which seeks to implement all of the ABC principles, whereas the eLabBench is special-purpose system for wet laboratory research in biology, which focuses primarily on activity-centric resource aggregation and activity roaming. Both technologies provides support for collaboration, but in two very different ways.

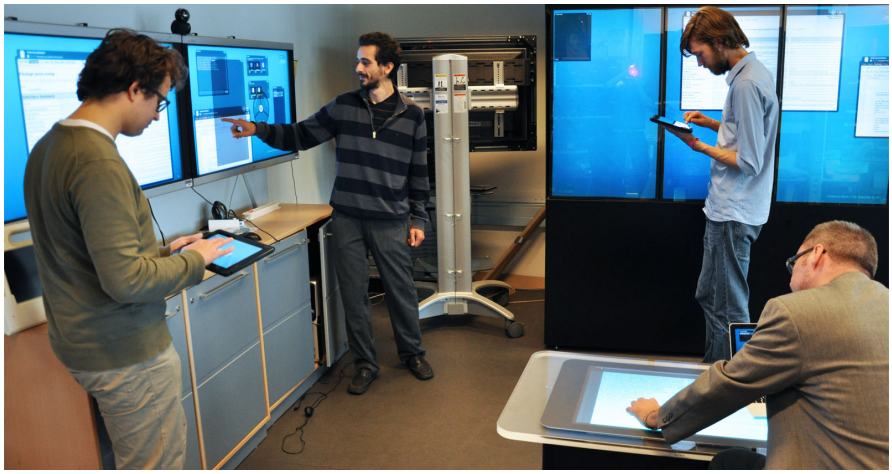


Fig. 3 ReticularSpaces in use by four users using six devices; two wall-based interactive displays, two mobile tablet computers, one tabletop display, and a laptop.

3.1 ReticularSpaces – Multi-device Collaborative Interactive Space

In personal computing one user is typically using one device in one location. A core challenge in the design of interactive space technology is to support work which is distributed across multiple devices, involving multiple users in multiple locations. ReticularSpaces [3, 4] was designed to address this challenge. Our approach was to use the 'activity' of a set of collaborating users as the core mechanism for coordination across multiple devices, users, and locations. Fig. 3 shows ReticularSpaces in use. By implementing the ABC principles, ReticularSpaces introduces a novel infrastructure and user interface for interactive spaces focusing specifically on activity-based support for device management, information management, mobility, and collaboration.

Following the ABC principle of **activity-centric resource aggregation**, the ActivityManager organizes all documents, resources, services, etc. into a set of activities. Each *activity* is composed of a set of *actions*, each again holding a set of *operations*. Each operation points to a *resource*, such as a document, a picture, html page, etc. Resources can also be external *services*, such as a device, like a printer, which can be accessed through an operation. Each activity has a list of *participants*, and only participants can access (resume/suspend) the activity, and its actions and operations. *Relationships* allows users to organize activities, actions, and operations in different workflow structures. Such structures could be simple association links showing which activities are related, as well as more complex workflow constraints specifying which activities has to be completed before another activity can be resumed. Fig. 4 (6) shows two related activities represented as a white line with text describing the type of relationship ('Belong to the same project').

The two main user interfaces of ReticularSpaces are the *Activity View* (Fig. 4) and the *Action View* (Fig. 5). The *Activity View* provides an overview of all relevant

Fig. 4 The *Activity View* showing a list of available activity managers (1), a list of users in this location (2), and the relevant set of activities from all mounted activity managers (3). Each activity (the white boxes) can be expanded to show its list of actions (4) and participants (5). Relationships between activities are shown as lines with a text label (6).

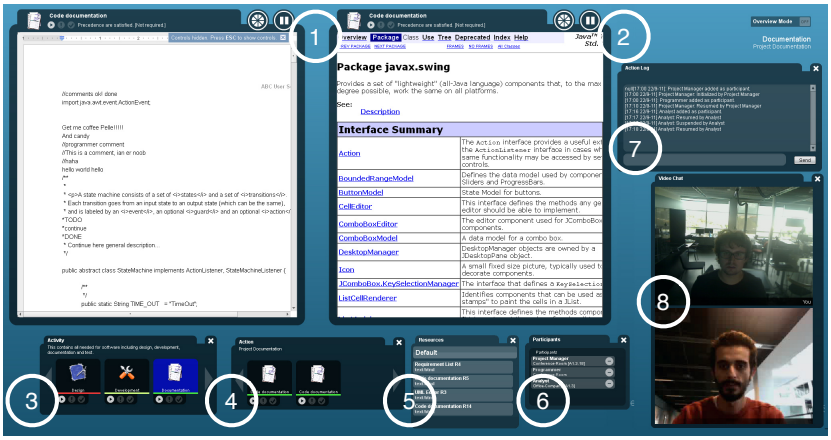


Fig. 5 The *Action View* is displayed when a user resumes an action by clicking on it in the activity view. The action view shows the action's operations and the resource each operation links to; in this case a source code document (1) and a web page showing Java documentation (2). The action view can show various overview panels as shown at the bottom of the view. From left to right these are overviews of; all actions in the overall activity (3); all operations in this action (4); available resources (5); and the participants (6). Users can communicate using the action log (7) and the remote video feeds (8).

activities from mounted activity managers, as well as contextual information about location, collocated users, and available activity managers. Each activity (the white box) can be expanded to show its list of actions and participants. Workflow relationships between activities are shown as lines with a text label. **Activity suspend and resume** in ReticularSpaces happens when a user clicks an action in the Activity

View (e.g. Fig.4 - 4) thereby resuming this action and is taken to the Action View. The *Action View* (Fig. 5) shows the action's operations and the resource each operation links to, such as a source code document (Fig.5 - 1) or a web page (Fig.5 - 2). The action view can show various overview panels as shown at the bottom of the view (Fig.5 - 3-6). These are overviews of: all actions in the overall activity (Fig.5 - 3); operations in this action (Fig.5 - 4); available resources (Fig.5 - 5); and the participants (Fig.5 - 6). Users switch between the Activity and Action Views by suspending and resuming an action. When clicking an action inside an activity in the Activity View, the action is resumed and the user interface shifts to show the Action View. When a 'suspend' button (not shown) is clicked in the Action View, the user interface shifts back to the Activity View.

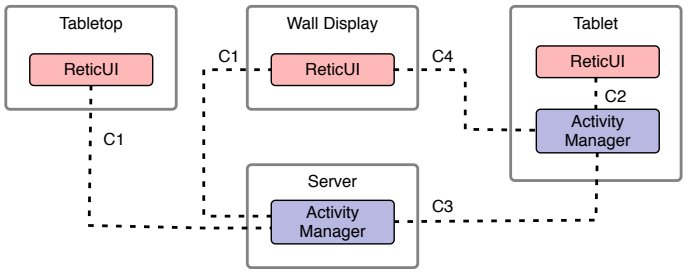


Fig. 6 The software architecture deployment diagram for the setup in Fig. 3.

In ReticularSpaces, the ABC principle of **activity roaming** is supported via a peer-to-peer (P2P) infrastructure that enables clients to manage their own activities or to discover and mount distributed activity managers. As illustrated in Fig. 6, the ReticularSpaces software architecture consists of two main components; the *ReticUI*, which is the user-interface component, and the *ActivityManager*, which stores, manages and distributes all data. Devices can run either ReticUI, the ActivityManager or both. Fig. 6 shows an deployment diagram reflecting the setup of devices and displays illustrated in the interactive space shown in Fig. 3. In this deployment setup, the interactive space runs a dedicated ActivityManager on a separate *Server*. Each of the fixed displays in the interactive space — i.e. the *Wall Display* and the *Tabletop* — runs a ReticUI client that connects to this central ActivityManager (the *C1* connections). When a mobile device – in this case a *Tablet* running its own ActivityManager connected to its ReticUI (the *C2* connection) — enters the interactive space, the two activity managers will discover and connect to each other (the *C3* connection). The ReticUI on the Wall Display can now mount the newly discovered ActivityManager to get access its activities and resources (the *C4* connection).

This architecture allows activities to be shared via a central ActivityManager (such as the *Server* in Fig. 6), thereby enabling users to access activities and their associated resources and data from distributed ReticUI clients. Moreover, the infras-

tructure supports a mobile ActivityManager (such as the *Tablet* in Fig. 6) to enter and be discovered by the interactive space. This allows displays inside the space to mount this newly discovered activity manager (as shown in Fig. 4(1)) and access the activities and data on the mobile device enabling a user to access and e.g. present data from this portable device on a display in the space. Vice versa, this also allows the ReticUI on the mobile device to discover and access data from the server in the space, thereby enabling mobile users to access and use local data. As such, the P2P architecture of ReticularSpaces has very flexible support for different mobility scenarios.

Data is managed as resources in the ReticularSpaces architecture, as also illustrated in Fig. 2. An ABC resource entity either contains the data, or points to a piece of data outside the ReticularSpaces architecture. For example, data are typically referenced using existing Internet protocols using URIs and a standardized protocol like HTTP, IMAP, or FTP, and are rendered based on their MIME type. Assuming that the ReticularSpaces runtime architecture have access to the Internet and thus online data resources, data will always be available during roaming between different devices and interactive spaces.

ReticularSpaces supports **activity sharing** and collaboration in multiple ways. Since an activity (and actions) has multiple participants, the activity and its data are shared and can be accessed by all participants. This enables a participant to resume an action, work on it in the Action View, and hence supports *asynchronous activity sharing* in which users can take turns in accessing and working with data in an activity and action. The P2P infrastructure allows participants to access data on shared and personal devices. Hence, users in the interactive space in Fig. 3 can access the activities and data in all activity managers inside the room, including the data on the tablet computer carried by the user entering the room. This allow for collocated collaboration and data sharing. Moreover, ReticularSpaces supports *synchronous activity sharing* where two (or more) participants resume the same action simultaneously on different devices. Synchronous collaboration is shown in the Action View in Fig. 5 in which two participant have resumed – and is hence engaged in – the same action. Synchronous action sharing have the effect that the user interface elements such as window positioning and size are synchronized in real time across the two device displays. This means that users working on different devices on the same action will see a synchronized view similar to the WYSIWIS principle[1]. ReticularSpaces also provides a live video feed between the two participating devices, as shown in Fig. 5. Finally, beside logging events (like resumption and suspension of actions), the *Action Log* allow users to type messages to each other. During asynchronous activity sharing, users can use the log to leave messages to each other, whereas during synchronous activity sharing, the log works as a instant messaging system.

[1] Acronym for "What I See Is What You See", used for groupware that guarantee that users see the same thing at all times.

Fig. 7 The eLabBench in use by a biologist conducting a lab experiment organized as an activity. Both physical resources, such as the test tube racks, as well as digital resources, such as research articles and a web-based lab notebook, are included in the activity.

3.2 eLabBench – Activity-based Collaboration for the Biology Laboratory

The eLabBench is part of an activity-based computing infrastructure for biology research [21, 22]. It is designed to support the transition between the planning, execution, and analysis phases of biologists' experimental research, by connecting desktop computers to interactive tabletops located in the wet-lab (see fig. 7 for an example of use). This work is highly collaborative. In the planning phase, graduate students will meet with post-docs or professors to discuss experiments, later plan their experimental protocols based on past experiments from colleagues. During the execution phase, multiple researchers can work on the same experiments either for teaching/learning purposes, or for time constraints, or based on unique expertise in a tool or method. Finally, during the analysis phase other participants can be involved, either to process material or run specific analyses. In practice, experiments are often iterated upon, until conclusive results are reached.

Activity-centric computing fits well with this kind of experiment-centric laboratory work, which on top of being collaborative, is highly distributed. Distribution can be global with research teams spread across the globe, and local at the scale of a laboratory building. Typically an experiment will be planned in meeting rooms and offices, conducted in a wet lab, with back and forth to specialized lab rooms with unique equipment. Once the data is gathered from several machines and servers, it will be analyzed in the office.

The eLabBench infrastructure allows biologists to conduct activities across multiple devices and location while supporting collaboration. Fig 8 shows the overall

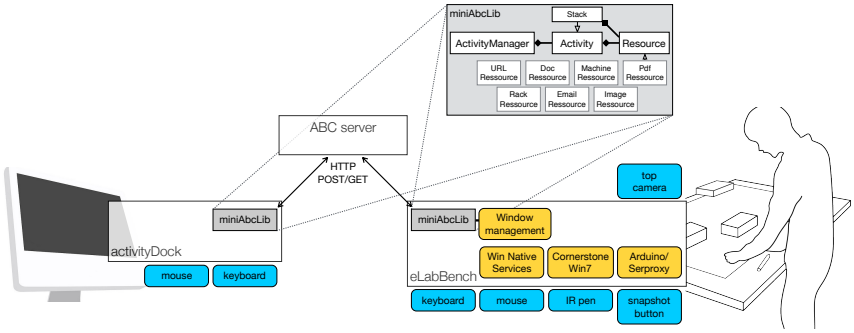


Fig. 8 The overall architecture of the eLabBench system, with its three main components: The activityDock running on Desktop computers, the ABC server in charge of roaming activities and the eLabBench running on tabletops. The activities are managed locally on each client with the miniAbcLib.

architecture of the eLabBench system. The *activityDock* is a desktop application running on personal computers (see also Fig. 10) and the *ABC Server* is a distributed data management infrastructure responsible for collecting and distributing digital data between personal computers and the eLabBenches. This architecture supports activity-centered resource aggregation, activity suspend/resume, activity roaming, and activity sharing.

Biologists typically organize their work and information around experiments, and an *experiment* is often the chosen unit for an activity. An activity is a collection or **aggregation of resources** that maps the digital information a biologist uses during the experimental cycle, and serves as a placeholder for all captured data. Biologists are able to create, delete and archive activities and their associated resources. Figure 9 shows a closeup picture of a biologist working at the eLabBench where he has resumed an activity containing relevant resources for this biology experiment. Examples of resources include a set of lab notes handled in a digital lab notebook (in this case a wiki-based system); article or videos explaining a specific protocol; online resources such as instructions on the use of biohazard material (in this case accessed through a web browser); or RNA and DNA sequences which are accessed, stored, and analyzed in a bioinformatics tool (in this case the CLC Bio Workbench[2]). More generally, the eLabBench supports the visualization of different kinds of digital content like PDF files, text documents, spreadsheets, pictures, web pages, emails, etc. By allowing biologists to access this broad range of digital content, the eLabBench aims at covering the most common information needs of a biologist.

By aggregating the relevant resources in a versatile structure, while leaving originals in their respective tools or system, i.e. email, bio-informatics suite, etc., the eLabBench enables the creation of reusable activities. The eLabBench also enables biologists to capture data while working at the bench during an experiment and adds it directly into the unfolding activity. This includes adding files to the activity,

[2] http://www.clcbio.com/products/clc-main-workbench/

Fig. 9 Activity-based Resource Aggregation — When the biologist conducts his experiment, he has access to all relevant resource and these are automatically shown on the eLabBench when resuming an activity. In this case, resources include an annotated testtube rack, the wiki-based lab notebook, and a set of other web sites. The menubar at the lower right allows the biologist to add additional resources to this activity, including websites, a calculator, and a video that records an experiment through the top-mounted camera.

such as pictures, documents, and spreadsheets, as well as handwritten notes, picture from the top-mounted camera, and digital annotations on physical racks of test tubes. Reusable activities coupled to capture tools enable long-term collaboration and reuse or activities.

Activity suspension and resumption is supported by an *Activity Bar* on the eLabBench allowing the user to access his or her list of activities and to resume these. The user is identified by a simple username/password login to the eLabBench. Only one activity at the time can be resumed on the eLabBench. When resuming an activity, the state of this activity is restored thereby bringing up the digital resources in the same state and UI position as when paused, just like a virtual desktop manager. By suspending and resuming activities from the activity bar, the user is able to alternate between many concurrently running experiments. It can also be a way to hand-off experiments to colleagues, once one is done its part.

The ABC Server supports **activity roaming** between an activityDock on a PC and the eLabBenches in the lab. This means that activities and resources can be moved between offices and laboratories, and in-between laboratories. Fig. 10 shows the use of the the *activityDock* in the office. The activityDock list all the activities that the current user participate in. Using the activityDock, a biologist can prepare an experiment in the office by creating an activity. Then, he can add resources to it, such as a protocol from the wiki lab book, PDF files of research papers, an email from a sample provider, etc. The biologist can also prepare racks from their offices by

Fig. 10 Activity Roaming — The *activityDock* in use by the biologist in his office while preparing for an experiment. He adds resources to the activityDock, such as the DNA structure shown on the right, which then later is accessed by resuming this activity on the eLabBench in the lab.

describing the layout and content of each tube, and thereby prepare for the physical execution of the experiment.

When moving to the laboratory, the biologist can access the relevant activities from any eLabBench in the lab. This allows him or her to load the necessary resources, access the experimental protocol, and record relevant information during the experiment. Documentation can be done with annotations to the protocol, by adding text notes to the activity, or directly in the wiki lab book. Activity roaming also enables the biologist to move an experiment between different eLabBenches, if need be. Back in the office, the biologist can resume the activity and thereby continue working on the activity and its resources. For example, checking notes and documenting more precisely the results of the experiments in the wiki lab book.

This basic roaming mechanism supports the iterative nature of biology work where biologists go back and forth between analytical work on the PC in the office and experimental work in the wet lab. It also supports distributed collaboration: while someone works in the laboratory on the experiment, a colleague can follow changes to the experiment by monitoring changes to its activity doc, but also provide support for instance by clarifying a shared protocol.

4 Activity-Based Collaboration for Interactive Spaces

Support for collaboration is core to interactive spaces; one of the main design rationales is to move beyond ‘personal computing’ towards ‘collaborative computing’. Opening up the collaborative design space, we can identify different types of collaboration that we would like to support in interactive spaces technologies. A simple taxonomy of such collaboration types is shown in Table 1. This taxonomy is di-

	Independently	*Engaged*	*Together*
Collocated	• Sharing Resources • Handing-over Tasks • Workspace Awareness	• Moving and Displaying Resources • Engaging in Tasks • Personalized Views	• Concurrent Modification of Resources • Simultaneous Task Management • Shared Views
Remote	• Turn-taking • Remote Awareness	• Remote Presentation • Communication	• Synchronized Views

Table 1 Simple taxonomy of collaboration in Interactive Spaces. The vertical *locality* dimension differentiates between users who are either collocated in one interactive space or located remotely in different interactive spaces. The horizontal *working* dimension differentiates between users working independently, focused, or closely together.

vided across the *locality* dimension – are users collocated or remote in one or more interactive spaces – and the *working* dimension – are users working independently, engaged or closely together. By '*engaged*' we mean when a group of users are engaged in the same activity but most of them are not actively working to change any resources. An example would be one person making a presentation for a group of people in the interactive space. In contrast, by '*together*' we mean when a group of users simultaneously work on the same resources and changing them. For example, pair-programming in software engineering or when two biologists sit in front of the same eLabBench and work on the same experiment. In the following, we will detail these six types of collaboration and how activity-based computing supports these, exemplified by the ReticularSpaces and eLabBench systems.

4.1 Independent Collaboration

The first type of collaboration is when users work collocated in an interactive space on independent work tasks. For example, two biologists working side-by-side in the laboratory on two separate eLabBenches. In this case, support for sharing of resources, handing over tasks, and collocated workspace awareness becomes relevant. In ABC, resources can be part of multiple activities, which allows for sharing of resources. For example, the two biologists can share both physical resources like a test tube rack, as well as digital resources, like a lab protocol (a 'URL Resource' in Fig. 2), between the two eLabBenches. If the two biologists are using the same protocol for different variations of an experiment, they can access, use, and update the information about the protocol and the description of a shared rack of test tubes that holds test material, from each of their eLabBenches.

Handing over tasks between users working inside an interactive space is a core feature. For example, different tasks are typically handed over and allocated during

stand-up meetings between a group of software engineers. In ABC, task allocation is supported by adding (and removing) users from the list of participants of an activity. ReticularSpaces was evaluated according to a software engineering scenario and this kind of task allocation was common for the participants to do. As for creating workspace awareness – i.e., enabling users inside the interactive space to monitor and see what others are doing and what is going on – the physical layout and design of the space and its interactive surfaces and devices plays a core role. The use of large-scale interactive displays on walls, tables, and tablet computers is instrumental in providing collocated users with a shared workspace awareness. For example, the large surface of the eLabBench allowed collocated biologists to monitor what was going on in the laboratory, including seeing the physical content and layout of the test tube racks on neighbor eLabBenches. Moreover, support for virtual workspace awareness is also needed in a collocated setup. For example, in ReticularSpaces the Activity View (Fig. 4) would continuously update the lists of available activity managers (i.e. devices) and users inside the interactive space, and the list of users would show what activity each user was engaged in. This provided users with a rudimentary workspace awareness about available devices (with activities and resources) and users inside the room.

The second type of collaboration is when users works remotely (e.g., in two different rooms) on independent tasks. For example, two software engineers working in separate offices. In this case, support or turn-taking and remote awareness become important. In ABC, turn-taking is supported by asynchronous collaboration, i.e., the ability that one participant can resume and continue working an activity, which has previously been suspended by another participant. In the eLabBench, this allows the supervisor to prepare an experiment for a group of students. This kind of turn-taking requires some sort of workflow support that enables the suspending participant to signal to the resuming participant that s/he can now take over. This kind of support for 'signaling' was, however, not designed as part of neither the eLabBench or ReticularSpaces, which was a shortcoming also discussed during the evaluation of them. Just like in collocated collaboration, workspace awareness is essential in remote situations. Remote awareness includes being aware of the location, activity, work load, and working context of collaborators, even when not directly collaborating with them on a task (right now). In ReticularSpaces remote awareness was supported through the same mechanisms as collocated awareness by showing the location and resumed activity for devices (activity managers) and users (see Fig. 4), whereas the eLabBench did not have any support for remote awareness.

4.2 Engaged Collaboration

The third type of collaboration is when users are engaged in the same activity in the same interactive space. For example, when a user enters an interactive space, gets access to a presentation on her portable device, and presents this on an interactive wall-display. In ABC this kind of engaged collocated collaboration is supported via

activity roaming, i.e., the ability to move an activity with its associated resources between devices, and activity adaptation, i.e., the adaptation of resources to the devices on which it is resumed. In ReticularSpaces, a user would be able to mount the activity manager on her portable device on the wall-display inside the room, and directly from the wall-display resume the relevant 'presentation' activity. This would then display the presentation adapted to the display size of the wall-display. Users inside the room can be added as participants to the presentation activity and hence get access to the presentation resources. Once participants are engaged in an activity, they would need support for personalized views, i.e., the ability to display, render, annotate, and change the resources of the activity. For example, users listening to a presentation might want to see a relevant video (which is a resource in the activity) on their own laptop or they might want to make personal notes to the presentation. Neither ReticularSpaces or the eLabBench supports this personalized view, which we found during our evaluations to be a limitation. For example, the annotations that can be made during an lab experiment in the eLabBench is stored as part of the activity and is hence available and editable for all participants of the activity; hence these annotation are not personal. Similarly, since the display and rendering of all resource are tightly synchronized in ReticularSpaces, if a participant would launch a video on her laptop, then this video would also be displayed on the wall-display for everyone to see. Our studies showed that better support for moving between personal and synchronized modes of collaboration needs to be investigated.

The fourth type of collaboration is when users are engaged in the same activity remotely. For example, when doing a presentation, which also involves remote participation. From an ABC point-of-view, remote collaboration is supported by the same means of activity roaming and activity adaptation, combined with synchronous activity sharing that allow for simultaneous access to resources on remotely located devices. In ReticularSpaces, remote participants could join the 'presentation' activity and the presentation would then be displayed on their local devices and the video-link and the action log chat window would be established to support communication across the two interactive spaces. Note that in this type of remotely engaged collaboration, the activity (i.e., the presentation) would still be driven from the presenter in one interactive space, having the remote participants joining as listeners. If the remote participants would start working on the activity (or more specifically on its resources), they would be working together instead (right-hand column in Table 1).

4.3 Collaborating Together

The fifth type of collaboration is when users collaborate actively together inside an interactive space. For example, when two biologist work together in front of the same eLabBench. In this type of collaboration, concurrent modification of shared resources is essential. For example, allowing the biologist to access, view, and modify the resources in the shared activity including updating the experimental protocol,

note down results, adjust content to the test tubes, and to add annotations to a pdf document. To a large extent, this was supported by the eLabBench system which was implemented using a multi-touch user interface that allowed several users (at least two) to work in front of it. However, the eLabBench system was limited by its underlying operating system (OS) – in this case Windows – and the applications running inside this OS. Hence, concurrent editing on the same device in the same application was not possible. For example, both biologists could not update information in the bioinformatics application on the eLabBench, and even though two browser windows each showing the web-based lab notebook could be accessed on the same eLabBench display by the two biologists, the limitation of single-input focus in the Windows OS was a limiting factor to true concurrent editing and browsing. Hence, to support true concurrent modification of resource in ABC, the underlying OS and applications used to access the resources need to support this concurrency — something which yet only exists in to very limited degree. The ReticularSpaces system was subject to the same limitations.

Collocated collaboration inside an interactive space also requires support for simultaneous task management allowing collaborating users to access, modify, and update task information. This would include updating basic task information, but in particular to update the list of participants who can work together on the task and the workflow relationships between tasks. In ReticularSpaces this was supported by having a very open access control mechanism; basically all participants of an activity could add participants (but only the user him or herself or the owner could remove participants). Moreover, workflow modeling was accessible for all participants of an activity. This allowed users to model simple workflows across existing activities while working on them inside the interactive space.

Finally, a shared view involving one or several interactive displays is a core requirement for collocated collaboration. Many interactive workspace technologies are designed to allow users to easily move and distribute resources across multiple shared displays inside the space (e.g., the i-LAND and iROS technologies allowed for this). In ABC there is less explicit support for this. Instead the more generic principles of activity roaming, activity adaptation, and activity sharing support creating a shared view across multiple devices and displays. In ReticularSpaces, for example, the same activity can be resumed on several displays inside an interactive space simultaneously, which will result in a synchronized view of all the resource on all displays. The benefit of this approach is that the same concepts and user interface mechanisms support both collocated and remote collaboration, since the system keeps a synchronized view on all resources, also when the displays are distributed in physically separate rooms. The drawback to this approach is that it is not possible to ‘split up’ an activity and display its resources on different displays. Hence, you would not be able to show e.g. a code section and an UML diagram from the same activity on two different displays, since all displays are kept synchronized.

In the sixth and final type of collaboration, users work closely together from two (or more) remote interactive spaces in separate locations. In this case, being able to share a synchronized view on the work task is essential. This is the principles of synchronous activity sharing in ABC as also implemented in ReticularSpaces.

The clear benefit to this solution is that collaborating users share the same view as they work closely together. For example, when two software engineers engage in pair-programming, they see the same code segment and the same viewport and zoom level of the UML diagram, which makes it easy to point to (with telepointers) and talk about the code. The drawback to the current implementation of the ReticularSpaces is the lack of more personalized views, as also discussed above. Hence, one of the engineers cannot open e.g., another UML diagram to consult without this diagram also being shown on the remote display of the other engineer.

5 Conclusion

This chapter offered an overview of the core concepts of Activity-Based Computing (ABC) with a special focus on the use of this approach for the design of interactive space technology. ABC builds on five core principles:

- *activity-centric aggregation* of computational resources which makes them easily accessible;
- support for *suspending and resuming* activities across multiple work context which supports multitasking and interruption;
- support for *roaming* activities (and their associated resources) between multiple devices, which supports mobility;
- *sharing* of activities (and resources) amongst multiple collaborating users; and
- enabling the activity to *adapt* to available resources and devices on which it is resumed and hence executed.

These five ABC principles were first outlined in 2002 and have proved to remain very stable over time when forming the basis for the implementation of different ABC technologies for different application areas, including the design of the interactive space technologies presented in this chapter. We presented two different types of technologies. *eLabBench* is an interactive space for wet lab biology work, including an interactive multitouch laboratory bench. In this setup, the eLabBench infrastructure implemented support for activity-centric resource aggregation, roaming, and sharing, which allow users inside and outside the biology lab to work together across multiple devices – the core requirements of an interactive space. The *ReticularSpaces* system was a much more elaborate infrastructure for interactive spaces, which implemented all of the ABC principles. This design case showed that the ABC principles are a very solid basis for the design of interactive spaces with a coherent computational and user interaction metaphor. As a general-purpose infrastructure for interactive spaces, ReticularSpaces have proved to be very flexible while providing many features in a simple and coherent manner.

The research on ABC have, however, also revealed a set of challenges still to be addressed. In particular, the core challenge to activity-centric computing is that most existing computer operating systems and applications do not have a notion

of ‘activity’, which means that it is notoriously difficult to implement an activity-centric computing model in contemporary operating systems and applications. A simple example is mail; a mail message is a typical resource in an activity where a set of related emails should be referenced by an activity. Technically this is rather straightforward using IMAP. However, there are – to our knowledge – no IMAP email client that allow rendering of the email message alone without the entire email client with all other emails visible – emails that are completely irrelevant to the current activity. Hence, a common challenge in the design and implementation of ABC technology have been that many applications had to be re-implemented in order to make the ‘activity-aware’. Data management has been another recurrent challenge to the implementation of ABC technologies. Some data types which typically reside on shared servers and can be accessed through standardized protocols is a perfect match for activity-centric data management since these can be accessed during activity roaming and suspend/resume. However, files that reside on personal computers have turned out to be a particular challenging to handle in ABC since such files (and folders) are hard to migrate or replicate across multiple devices. One approach to pursue here is adopt file replication mechanisms as implemented in Dropbox and similar file synchronization protocols. But again, these protocols are agnostic to the concept of activity, and activity-centric data management and replication thus has to be implemented in addition to the basic file synchronization.

Based on this, we can conclude that Activity-Based Computing and its principles seems to be a technological approach very well suited for the design and implementation of interactive spaces. However, ABC and interactive space technology have proven to be particular hard to implement on top on exiting personal computing operating systems, applications and file systems, which have no notion of the activities users are engaged in. Therefore, we expect that a real well-functioning operating system for interactive spaces cannot be built from existing operating systems and therefore a new generation of operating systems are needed. We would argue that such an operating system could benefit from incorporating support for activity-based computing.

References

1. Bannon, L., Cypher, A., Greenspan, S., Monty, M.L.: Evaluation and Analysis of Users’ Activity Organization. In: CHI’83: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 54–57. ACM (1983)
2. Bardram, J., Doryab, A., Gueddana, S.: Activity-Based Computing–Metaphors and Technologies for Distributed User Interfaces. In: Distributed User Interfaces, pp. 67–74. Springer (2011)
3. Bardram, J., Gueddana, S., Houben, S., Nielsen, S.: ReticularSpaces: Activity-based Computing Support for Physically Distributed and Collaborative Smart Spaces. In: Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems, CHI ’12, pp. 2845–2854. ACM, New York, NY, USA (2012). DOI 10.1145/2207676.2208689
4. Bardram, J., Houben, S., Nielsen, S., Gueddana, S.: The Design and Architecture of ReticularSpaces: An Activity-based Computing Framework for Distributed and Collaborative

Smartspaces. In: Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems, pp. 269–274. ACM (2012)
5. Bardram, J.E.: Activity-based Computing for Medical Work in Hospitals. ACM Transactions on Computer-Human Interaction **16**(2), 1–36 (2009)
6. Bardram, J.E., Bunde-Pedersen, J., Doryab, A., Sørensen, S.: CLINICAL SURFACES: Activity-Based Computing for Distributed Multi-Display Environments in Hospitals. In: INTERACT'09: Proceedings of the Conference on Human Computer Interaction, *Lecture Notes in Computer Science*, vol. 5727, pp. 704–717. Springer (2009)
7. Bardram, J.E., Jeuris, S., Houben, S.: Activity-Based Computing: Computational Management of Activities Reflecting Human Intention. AI Magazine **36**(2), 63–72 (2015)
8. Bergman, O., Beyth-Marom, R., Nachmias, R.: The Project Fragmentation Problem in Personal Information Management. In: CHI'06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 271–274. ACM (2006)
9. Biehl, J.T., Baker, W.T., Bailey, B.P., Tan, D.S., Inkpen, K.M., Czerwinski, M.: Impromptu: A New Interaction Framework for Supporting Collaboration in Multiple Display Environments and its Field Evaluation for Co-located Software Development. In: CHI'08: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 939–948. ACM, New York, NY, USA (2008)
10. Boardman, R., Sasse, M.A.: "Stuff Goes Into the Computer and Doesn't Come Out": A Cross-tool Study of Personal Information Management. In: CHI'04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 583–590. ACM (2004)
11. Christensen, H.B., Bardram, J.E.: Supporting Human Activities - Exploring Activity-Centered Computing. In: UBICOMP'02: Proceedings of the International Conference on Ubiquitous Computing, *Lecture Notes in Computer Science*, vol. 2498, pp. 107–116. Springer (2002)
12. Gonzalez, V.M., Mark, G.: "Constant, Constant, Multi-tasking Craziness": Managing Multiple Working Spheres. In: CHI'04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 113–120. ACM (2004)
13. Henderson Jr., D.A., Card, S.: 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 (1986)
14. Houben, S., Tell, P., Bardram, J.E.: ActivitySpace: Managing Device Ecologies in an Activity-Centric Configuration Space. In: Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces, ITS '14, pp. 119–128. ACM, New York, NY, USA (2014)
15. Johanson, B., Fox, A., Winograd, T.: The interactive workspaces project: Experiences with ubiquitous computing rooms. IEEE Pervasive Computing **1**, 67–74 (2002)
16. Johanson, B., Hutchins, G., Winograd, T., Stone, M.: Pointright: experience with flexible input redirection in interactive workspaces. In: UIST'02: Proceedings of the SIGCHI Conference on User Interface Software Technology, pp. 227–234. ACM, New York, NY, USA (2002)
17. MacIntyre, B., Mynatt, E.D., Voida, S., Hansen, K.M., Tullio, J., Corso, G.M.: Support for multitasking and background awareness using interactive peripheral displays. In: Proceedings of the 14th annual ACM symposium on User interface software and technology, pp. 41–50. ACM (2001)
18. Miyata, Y., Norman, D.A.: Psychological issues in support of multiple activities. In: D.A. Norman, S.W. Draper (eds.) User centered system design: New perspectives on human-computer interaction, chap. 13, pp. 265–284. Lawrence Erlbaum Associates, Hillsdale, NJ (1986)
19. Roseman, M., Greenberg, S.: Building real-time groupware with groupkit, a groupware toolkit. ACM Transactions on Computer-Human Interaction (TOCHI) **3**(1), 66–106 (1996)
20. Streitz, N.A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., Steinmetz, R.: i-LAND: an Interactive Landscape for Creativity and Innovation. In: CHI'99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 120–127. ACM, New York, NY, USA (1999)
21. Tabard, A., Hincapi Ramos, J.D., Bardram, J.: The elabbench in the wild: supporting exploration in a molecular biology lab. In: Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems, CHI '12, pp. 3051–3060. ACM, New York, NY, USA (2012). DOI 10.1145/2207676.2208718

22. Tabard, A., Hincapi-Ramos, J.D., Esbensen, M., Bardram, J.E.: The elabbench: An interactive tabletop system for the biology laboratory. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces 2011, ITS2011, pp. 301–310. ACM, New York, NY, USA (2011)
23. Weiser, M., Gold, R., Brown, J.S.: The origins of ubiquitous computing research at parc in the late 1980s. IBM systems journal **38**(4), 693 (1999)
24. Winograd, T., Flores, F.: Understanding computers and cognition: A new foundation for design. Intellect Books (1986)