The Design and Architecture of ReticularSpaces – an Activity-Based Computing Framework for Distributed and Collaborative SmartSpaces

Jakob E. Bardram, Steven Houben, Sofiane Gueddana, Søren Nielsen
The Pervasive Interaction Technology Laboratory
IT University of Copenhagen, Rued Langgaardsvej 7, DK-2300 Copenhagen, Denmark
{bardram,shou,sgue,snielsen}@itu.dk

ABSTRACT

Interactive workspaces are increasingly physically distributed, highlighting the challenge of building interfaces that support group interaction with digital documents through multiple locations and devices. This paper presents the technical implementation and user interface of *ReticularSpaces*. Based on the concepts and principles of Activity-Based Computing (ABC), *ReticularSpaces* implements a novel approach to smart space user interfaces, and supports task-based information management, mobility, and collaboration.

Author Keywords

Distributed User Interfaces, Multiple Display Environments, Smart Spaces, Activity-Based Computing

ACM Classification Keywords

H.5.2 INFORMATION INTERFACES AND PRESENTATION: User Interfaces—*User interface management systems*

INTRODUCTION

Since the pioneering work at Xerox PARC on the design of the Ubiquitous Computing technologies for meeting support [17], a significant line of research has been investigating such kind of 'smart space' technologies. 'Smart space' is an ill-defined concept, but in general the term covers research into augmenting a physical space – typically a meeting room – with computational resources. These resources often include; interactive vertical and horizontal collocated multi-touch displays of various sizes; mobile devices including laptop computers, tablet computers, and smart phones; embedded sensors and room control for sensing of people and objects in the room, and control of e.g. lighting. A core research problem – often referred to as 'Distributed User Interfaces' – is investigating how the devices and displays (fixed and mobile) can work together in a unified and interlinked way. Another research problem is how to design a smart space infrastructure for device discovery, interoperation, and information management.

EICS'12, June 25–26, 2012, Copenhagen, Denmark.

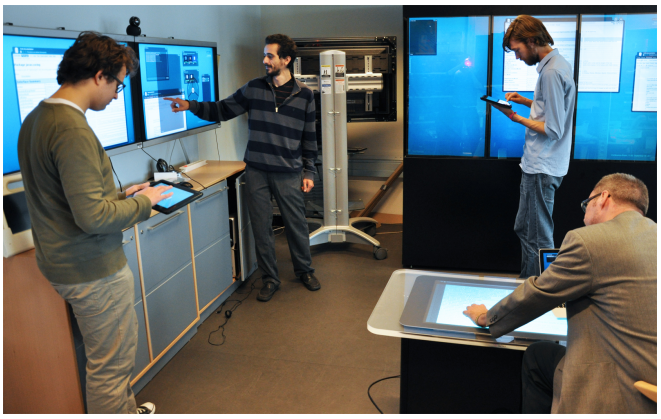

Figure 1. The RecticularSpaces smart space setup, comprising of large wall-based displays, horizontal tabletop displays, laptops, and tablet computers.

Looking at recent research into these problems, some of the important design tradeoffs in the design of smart space technologies seem to be:

- Operating System (OS) – should a smart space run on existing (primarily commercial) personal computing OSs like Windows, Linux, and Mac OS? Or is there a need for a new type of operating system for smart space technologies?
- User Interface (UI) – should smart space technology extend and build on existing user interfaces (mostly designed according to the desktop metaphor)? Or is a novel design metaphor and user interface needed?
- Information Management – does the smart space provide support for 'smart' or context-aware information management (like automatic capture and access of relevant documents for a meeting)? Or is information management left for the users to handle?
- Mobility – to what degree should a smart space support mobile or nomadic users? Does smart space support extend beyond the smart space itself? Should users be able to enter and leave the smart space?
- Devices – what sort of devices are supported by the smart space? Does it support fixed wall displays? Mobile devices? Phones? Service Devices such as printers and scanners? Embedded devices like HVAC controls, sensors, and lighting?

- Collaboration – what sort of collaboration is supported? Collocated collaboration within the smart space? Or remote collaboration with users outside the smart space?

The work done on the Ubiquitous Computing environment [17] and the iLand design of RoomWare technologies [15] took a rather fundamental approach by designing and building special-purpose hardware, displays, infrastructures, operating systems, and even furniture for smart spaces. The core argument was, that the existing approach to computing, designed according to the personal computer model with its desktop metaphor UI, was not suited for this new kind of 'Post-PC' computing environments. And hence, a new technology stack including everything from hardware to UI was needed. More recent work on smart space technologies have to a larger degree taken the existing OSs and UIs for granted, and have tried to extend these with support for smart spaces. For example, the GAIA meta-operating system [12] and the One.World framework [11] have been extending contemporary operating systems to also support features like application migration and service discovery, which are used in a smart space. Similarly, recent approaches like iRos [13], ComPUTE [6], Impromptu [8], Aris [7], and AirLift [1] focus on extending and augmenting the UI of existing personal computing operating systems with features for smart space interaction. This include graphics redirection; pointer, control, and view redirection; annexation of local displays; relocation of information across displays; and engagement of multiple users in collaboration on shared content.

Hence, prior research have investigated a wide range of issues and have provided a set of infrastructure and UI technologies for smart spaces. However, a set of the core questions listed above still seems rather unexplored. First, even though much of the recent research have taken the approach to merely extending the contemporary UI desktop metaphor, it is still questionable if a 'desk' with its icons, folders, files, and documents is the right design metaphor for a smart space. Beyond the obvious familiarity to users, this metaphor seems to provide little leverage for a smart space. Second, even though most research incorporate support for mobile devices entering and leaving a smart space, this support rarely goes beyond the room itself. But mobile and nomadic work is increasingly an important target for computer technologies and hence smart spaces. Smart spaces should not only be targeting meeting rooms in regular office buildings, but will also be use in e.g. hospitals [4] and biology laboratories [11], which has a high degree of mobile and nomadic work patterns. Third, few approaches support collaboration beyond collocated collaboration inside the smart room. But an important part of work *inside* a smart room is to be able to collaborate with users *outside* it. Finally, existing research have been addressing information management by focusing on document sharing and context-aware file management. There is little or no support for the overall tasks that users are engaged in, and the 'smart' space is often surprisingly unaware of what activities and tasks are taking place inside it.

In sum, there is still a set of challenges in the design of smart space technology, infrastructures, and interfaces that support spanning activities over a large amount of devices, better support for content and document management, while supporting collaboration amongst multiple users and across multiple locations[1].

In this paper we present *ReticularSpaces*, which is novel infrastructure and user interface for smart spaces [2]. Figure 1 shows *ReticularSpaces* in use. Compared to prior research on smart spaces, *ReticularSpaces* focuses specifically on providing answers to 4 of the questions listed above:

- **User Interface** – *ReticularSpaces* implements a completely novel UI metaphor for smart space. As such, *ReticularSpaces* does not rely or expand on existing desktop UIs, but implement a new design based on an 'Activity' metaphor. The goal is to align the UI metaphor to the activities of the users, rather than their desktop.
- **Information Management** – according to the 'Activity' metaphor, *ReticularSpaces* implements an activity-based computing approach to information and data management. Thus, in contrast to the desktop metaphor where information management is modeled according to an office (files, folders, trashcans), the goal of *ReticularSpaces* is to align information management to the activities of the users.
- **Mobility** – *ReticularSpaces* has inherent support for mobility of users as well as devices. In addition to supporting devices entering and leaving a smart room, *ReticularSpaces* supports mobility within a larger smart space, which typically spans a large geographical areas like a hospital, large factory, or a campus.
- **Collaboration** – *ReticularSpaces* has inherent support for collaboration amongst users and devices. In addition to supporting collocated collaboration by display sharing inside a room, remote collaboration across a larger site is supported.

This paper presents the technical design and implementation of the *ReticularSpaces* system infrastructure and UI management system. The overall motivation, design, and usage of *ReticularSpaces* is described in [2].

RETICULARSPACES TECHNICAL ARCHITECTURE

ReticularSpaces is the 5th generation of our research into activity-based computing (ABC) and applies a peer-to-peer (P2P) architecture illustrated in Figure 2. In contrast to the 3rd [3] and 4th [5] generation of the ABC infrastructure where the *Activity Manager* was located on a central server, each peer (client) in *ReticularSpaces* has an instance of the activity manager running. Each client can also run the user-interface client called *ReticUI* . A *ReticUI* client knows the network address (default is 'localhost') of its activity manager (connection of type C1 in Figure 2). Activity managers are constantly looking for other activity managers (using mDNS), and can thereby discover and connect to each other (C2). An *ReticUI* client can connect to any activity manager which

[1]This conclusion is in line with the discussion and conclusion from recent CHI workshops on the design of smart space and related systems. For example the CHI 2006 workshop on 'Information Visualization and Interaction Techniques for Collaboration across Multiple Displays' [16] and the CHI 2011 workshop on 'Distributed User Interfaces' [10]

is discovered and enlisted by its (local) Activity Manager. Hence, in Figure 2, the activity browser on *client_1* can mount and access data in activity manager on *client_2* (C3), once the activity manager of *client_2* has discovered and connected to the activity manager of *client_1* (C2).

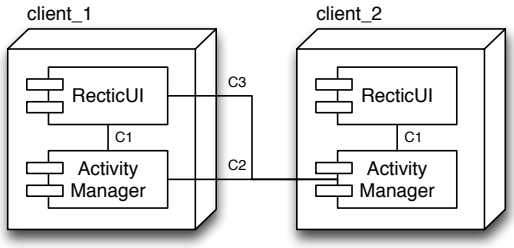

Figure 2. A schematic view of the *ReticularSpaces* P2P architecture.

A more elaborate deployment example is illustrated in Figure 3. This deployment setup is the demonstration setup illustrated in Figure 1 and used for the evaluation presented in [2]. In Figure 3 the smart room runs a dedicated smart room activity manger on a separate server machine. The *ReticUI* client of each of the displays fixed inside the smart room – such as the wall and tabletop displays – connects to this central activity manager (C1 type of connections). When a mobile device (in this case the tabletPC) enters the smart space network environment, the two activity managers will discover and connect to each others (C2). The *ReticUI* client will display available activity managers as shown in Figure 4. By clicking on the label for an activity manager, the user 'mounts' this activity manager and thereby establish connection C3.

This illustrates the technical setup in the scenario where a user enters the smart room with a tablet computer, which is discovered and displayed on a wall display. Any user can then go to the wall display, mount the activity manager, and access its data from the wall display.

Activity Manager

The core component in the ABC infrastructure is the Activity Manager. Figure 5 illustrates the overall software architecture of the activity manager. The activity manager is implemented using the Aexo framework [9] by extending the base Aexo component called `AComponent`. An Aexo component is hosted by the Aexo runtime on a host device, and uses an event-based RESTful communication interface. This allows clients to get, post, and delete data as well as subscribe to changes in data in a component, in which case the client (subscriber) will get notified if data is updated. In *ReticularSpaces* , the Aexo `ActivityManagerComponent` basically initializes itself within the Aexo runtime system, and then holds a reference to the `ActivityController`, which is responsible for all the logic in the ABC infrastructure. Using three registries, the activity manager holds references to other discovered activity managers, services (such as printers, sensors, etc.), and attached *ReticUI* clients.

Activity managers maintain information about the physical location of devices running a *ReticUI* client. In the example above, the smart room server's activity manager knows

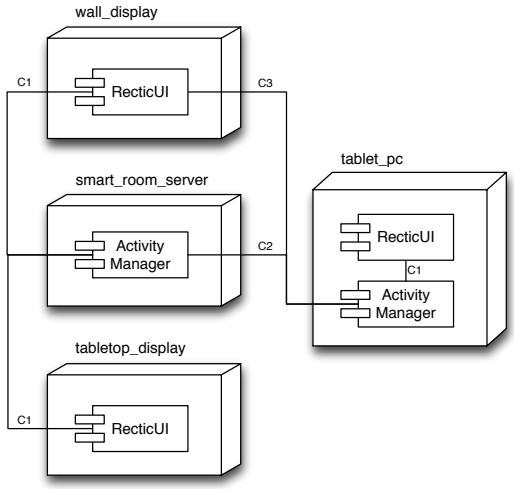

Figure 3. The deployment diagram for the setup in Figure 1.

Figure 4. Details from the Activity View of *ReticUI* (Figure 8) showing a list of activity managers available in this location (the 'pITLab'). Each activity manager has a color code, which is used to identify activities from this activity manager.

the location of the wall and tabletop displays, as well as the location of the tablet computer.

Information Management

The activity manager's data model is shown in Figure 6. This model organize all documents, resources, services, etc. that are relevant for a human activity into a corresponding 'Computational Activity', or just 'Activity'. 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 done before another activity can be resumed. Because of the overlapping properties between activities, actions and operations, they are abstracted into an *enactment*.

Dynamic Behavior

From a dynamic view, *ReticularSpaces* implements a distributed model-view-controller (dMVC) architecture. The activity manager holds the activity model, which is replicated

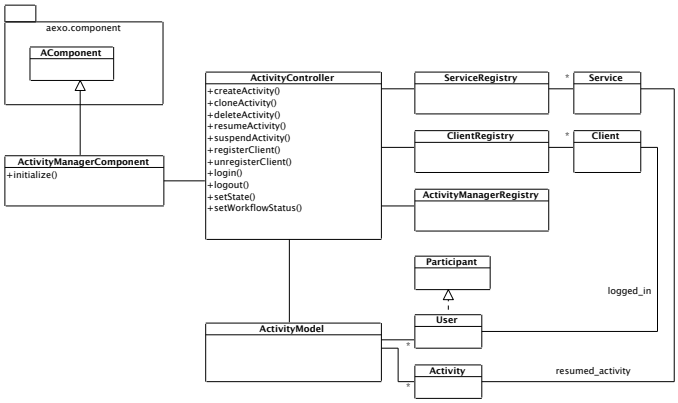

Figure 5. The Activity Manager Class Diagram.

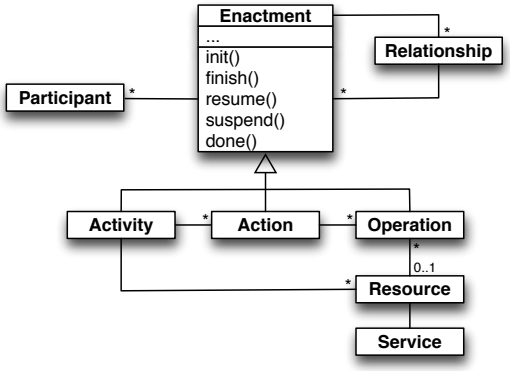

Figure 6. The ABC Ontology used in *ReticularSpaces*.

to each *ReticUI* client. On each *ReticUI* client, a standard (local) MVC patterns keeps views updated, and manages changes to the model. Data model replication and synchronization is done using the event-based system in Aexo.

When a *ReticUI* client connects to the infrastructure it can display and manipulate data elements like activities, actions, operations, resources, and services. Changes to the client's replicated data model propagate to the distributed model and updates other client models. To render resources and services, *ReticUI* clients access an URI of the service and show any data coming from that service, with a different rendering component according to the data type.

This distributed model-view-controller pattern is also used in the collaborative features of *ReticularSpaces*. Since, the layout and location of certain UI elements is part of the data stored in the data model, the distributed model-view-controller setup implies that the view of certain relevant part of the *ReticUI* user interface is synchronized in real time across multiple displays. This means that users working on different devices on the same activity will see a synchronized view similar to the WYSIWIS principle[2].

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

Figure 7. Use of *ReticUI* on very large wall displays (top) and on mobile devices in remote locations (bottom).

RETICULARSPACES USER INTERFACE

As described above, *ReticularSpaces* has adopted a completely new UI metaphor based on the principles of activity-based computing [2]. This UI – called *ReticUI* – is designed to run on a wide range of interactive displays, and supports different display features:

Size – *ReticUI* implements panning and zooming on an indefinite canvas, and hence scales from small tablet displays (12"), to tabletop displays (46"), to very large wall displays (120" or more) – see Figure 7. In contrast to e.g. Exposé in the Mac OS, or ScalableFabric [14], *ReticUI* widgets and components can be interacted with in any scale.

Touch modality – *ReticUI* supports both pen-based interaction, single-touch, and multi-touch. Depending on the type of touch, various gestures implements different commands, like zooming, panning, and action resume/suspend.

Video – if devices have a video camera, video connections are automatically established during activity sharing session of remotely located devices.

Commands – all commands in *ReticUI* are accessible through gestures and/or pie menus. Since *ReticUI* can run on very large displays, pie menus can be activated by clicking the background canvas in any place where the user may be located in front of the display.

The UI of *ReticUI* is shown in Figure 8 and 9. The main 'widgets' in the activity-based computing UI metaphor are; activities; actions; operations; resources; services and participants. These widgets are organized according to the activity ontology shown in Figure 6. *ReticUI* consists of two main

Figure 8. The *Activity View* showing a list of available activity managers, a list of users in this location, and the relevant set of activities from all mounted activity managers. Each activity (the while box) can be expanded to show its list of actions and participants. Relationships between activities are shown as lines with a text label.

Figure 9. 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 and a web page showing Java documentation. 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; all operations in this action; available resources; and the participants. On the right side the collaboration windows are shown, including (from the top); the action log and the remote video feeds. Users can switch between the activity and action views by suspending and resuming an action.

views; the *Activity View* (Figure 8) and the *Action View*. The activity view provides an overview of all relevant activities from mounted activity managers, as well as contextual information about location, collocated users, and available activity managers. Each activity (the while box) can be expanded to show its list of actions and participants. Workflow relationships between activities are shown as lines with a text label. The *Action View* (Figure 9) is displayed when a user resumes an action by clicking it in the activity view. The action view shows the action's operations and the resource each operation

links to, such as a text document or a web page. 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; operations in this action; available resources; and the participants. On the right side the collaboration windows are shown, including (from the top); the action log (which works as a instant messaging system when users are engaged in online activity sharing) and the remote video feeds. Users can switch between the activity and action views by suspending and resuming an action.

SUMMARY

This paper has presented the software architecture and user interface technology of *ReticularSpaces*. *ReticularSpaces* is a smart space technology based on the principles of activity-based computing. The core design goal of *ReticularSpaces* was to provide a novel user interface for smart space technologies based on 'activity' as the core metaphor; to support task-based information management across several displays and locations; to support mobility inside a larger smart space such as large factories, hospitals, or a campus; and to support collaboration both collocated inside a smart room, as well as between remotely distant locations. We are currently working on improving the *ReticularSpaces* technologies. On the infrastructure level, a more robust and scalable infrastructure is being designed by leveraging cloud technologies. On the interface part, better support for learning this new UI metaphor is needed and the workflow visualizations and usage needs to be improved. The long term goal is to use *ReticularSpaces* for projects within different domains, and for real-world deployment.

REFERENCES

1. T. Bader, A. Heck, and J. Beyerer. Lift-and-drop: crossing boundaries in a multi-display environment by airlift. In *Proc. of AVI 2010*, pages 139–146. ACM, 2010.
2. J. Bardram, S. Gueddana, S. Houben, and S. Nielsen. Reticularspaces: Activity-based computing support for physically distributed and collaborative smart spaces. In *Proc. of CHI 2012*. ACM, 2012.
3. J. E. Bardram. Activity-based computing for medical work in hospitals. *ACM Transactions on Computer-Human Interaction*, 16(2):1–36, 2009.
4. J. E. Bardram, J. Bunde-Pedersen, A. Doryab, and S. Sørensen. Clinical surfaces — activity-based computing for distributed multi-display environments in hospitals. In *Proc. of INTERACT 2009*, pages 704–717. Springer-Verlag, 2009.
5. J. E. Bardram, J. Bunde-Pedersen, and M. Soegaard. Support for activity-based computing in a personal computing operating system. In *Proc. of CHI 2006*, pages 211–220, New York, NY, USA, 2006. ACM Press.
6. J. E. Bardram, C. Fuglsang, and S. C. Pedersen. Compute: a runtime infrastructure for device composition. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, pages 111–118. ACM, 2010.
7. J. T. Biehl and B. P. Bailey. Aris: an interface for application relocation in an interactive space. In *Proc. of GI 2004*, pages 107–116, 2004.
8. J. T. Biehl, W. T. Baker, B. P. Bailey, D. S. Tan, K. M. Inkpen, and M. Czerwinski. Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development. In *Proc. of CHI 2008*, pages 939–948. ACM, 2008.
9. J. Bunde-Pedersen. *Distributed Interaction for Activity-Based Computing*. PhD thesis, Computer Science Department, University of Aarhus, Denmark, 2009.
10. J. A. Gallud, R. Tesoriero, J. Vanderdonckt, M. Lozano, V. Penichet, and F. Botella. Distributed user interfaces. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, CHI EA '11, pages 2429–2432. ACM, 2011.
11. R. Grimm. One.world: Experiences with a pervasive computing architecture. *IEEE Pervasive Computing*, 3(3):22–30, July 2004.
12. C. K. Hess, M. Román, and R. H. Campbell. Building applications for ubiquitous computing environments. In *Proc. of Pervasive 2002*, pages 16–29. Springer-Verlag, 2002.
13. B. Johanson, A. Fox, and T. Winograd. The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1:67–74, April 2002.
14. G. Robertson, E. Horvitz, M. Czerwinski, P. Baudisch, D. R. Hutchings, B. Meyers, D. Robbins, and G. Smith. Scalable fabric: flexible task management. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '04, pages 85–89. ACM, 2004.
15. N. A. Streitz, J. Geißler, T. Holmer, S. Konomi, C. Müller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. i-land: an interactive landscape for creativity and innovation. In *Proceed. of CHI 1999*, pages 120–127. ACM, 1999.
16. L. Terrenghi, R. May, P. Baudisch, W. MacKay, F. Paternò, J. Thomas, and M. Billinghurst. Information visualization and interaction techniques for collaboration across multiple displays. In *Extended Proc. of CHI 2006*, CHI '06, pages 1643–1646. ACM, 2006.
17. M. Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):66–75, September 1991.