Activity-Centric Support for Ad Hoc Knowledge Work – A Case Study of co-Activity Manager

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

Jo Vermeulen, Kris Luyten, Karin Coninx
Expertise Centre for Digital Media
Hasselt University – tUL – iMinds,
Wetenschapspark 2, B-3590 Diepenbeek, BE
firstname.lastname@uhasselt.be

ABSTRACT
Modern knowledge work consists of both individual and highly collaborative activities that are typically composed of a number of configuration, coordination and articulation processes. The desktop interface today, however, provides very little support for these processes and rather forces knowledge workers to adapt to the technology. We introduce co-Activity Manager, an activity-centric desktop system that (i) provides tools for ad hoc dynamic *configuration* of a desktop working context, (ii) supports both explicit and implicit *articulation* of ongoing work through a built-in collaboration manager and (iii) provides the means to *coordinate* and share working context with other users and devices. In this paper, we discuss the activity theory informed design of co-Activity Manager and report on a 14 day field deployment in a multidisciplinary software development team. The study showed that the activity-centric workspace supports different individual and collaborative work configuration practices and that activity-centric collaboration is a two-phase process consisting of an activity sharing and per-activity coordination phase.

Author Keywords
Activity Theory, Desktop Interface, Activity-Centric Computing, Collaborative Work

ACM Classification Keywords
H.5.2 [Information interfaces and presentation]:User Interfaces. - Graphical user interfaces.

INTRODUCTION
Knowledge work is typically composed of both *individual* and highly *collaborative* work. This means that knowledge workers use personal computing devices to perform individual tasks and activities that are part of a larger collaborative working context. The individual work is in many cases dependent on and driven by information that is provided by coworkers or other stakeholders that are part of the overall activities. There is thus a high demand for collaboration amongst co-workers which results in an increased level of project fragmentation due to the large number of tasks and activities one typically performs at the same time [7, 11, 25]. Nevertheless, despite this highly collaborative nature of knowledge work, many users still want to tailor their part of the work to their personal preferences.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CHI 2013, April 27–May 2, 2013, Paris, France.
Copyright 2013 ACM 978-1-4503-1899-0/13/04...$15.00.

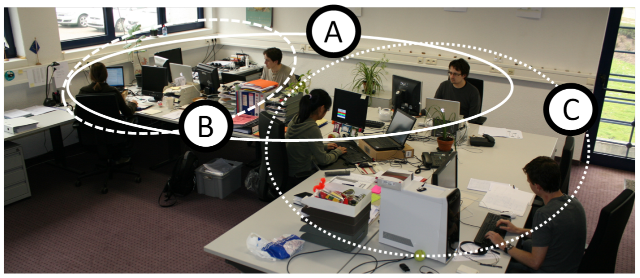


Figure 1. A multidisciplinary software development team. The groups (a,b,c) symbolize some of the collaborative relations between the members.

Figure 1 shows a multidisciplinary software development team. This team collaborates on different projects with different colleagues in partially overlapping subgroups (marked as *a*, *b* and *c*). For these projects the team members continuously collaborate on shared parallel activities, which require some form of coordination (e.g. sharing files and being aware of each other's updates on those files) between individually performed work.

Current desktop interfaces, however, provide very little support for this type of activity-centric collaboration. Although many stand-alone tools are available to users, studies have shown that these are frequently the source of interruptions and project fragmentation because they are disconnected from other tools and functions, which are used in the same activity [10, 18, 21] .

In general, research has pointed out that there is a fundamental mismatch between the design and functionality of modern desktop systems and the need for more activity-oriented support in a work setting [5, 23]. As we will discuss in the 'Related Work' section, many of the proposed solutions focus very much on the individual interaction with the desktop interface, often minimizing the importance of the collaborative aspect of knowledge work.

This paper introduces *co-Activity Manager* (cAM), an activity-centric multi-user desktop manager integrated with the Windows 7 operating system. co-Activity Manager was specifically designed to facilitate collaboration by supporting:

1. **Activity-centric desktop management** to minimize the configuration work for different parallel activities;
2. **Activity sharing** that allows users to share and deploy activities in a collaborative multi-device setup; and
3. **Activity-centric collaboration tools** (including file sharing and messaging) thereby moving communication channels into the activity abstraction.

We present the design and implementation of co-Activity Manager, report on a 14 day field deployment in a multi disciplinary software development team, and discuss the lessons learned from the implementation and deployment of the system.

RELATED WORK

The earliest approach to update the desktop interface was the introduction of virtual workspaces in the Rooms system [17]. There are, however, many other systems and applications that use virtual workspaces as an approach to solve some of the problems of the desktop interface. Systems such as Groupbar [33], Quickspace [20] and Task Gallery [31] use the desktop workspace in different ways. Unfortunately, although virtual desktops partly solve the workspace shortage, they increase the cognitive load when more than four desktops are used, as users must remember themselves what information is located in which virtual desktop [30].

Amongst others, Kaptelinin and Czerwinski [23] summarized a number of novel approaches that have been proposed to restructure the desktop interface. These novel approaches are based on fundamental concepts such as time (e.g., LifeStreams [15]), the relation between information (e.g., Haystack [1]), physics (e.g., Bumptop [2]), virtual workspaces (e.g., Rooms [17] and GroupBar [4, 33]), tasks (e.g., TaskTracer [8]) or activities (e.g., ABC [4], Giornata [35], CAAD [29] and Umea [22]). In this paper, we follow the latter approach.

Several approaches have introduced higher-level structures in the form of tasks or activities. Taskmaster [6], e.g. uses a traditional email layout to organize activities. This implementation is very helpful as a tool to enhance communication and to keep track of work flow but is too limiting as the users can only use a set of pre-defined objects inside the application rather than their normal desktop workspace. Some solutions try to reduce the mental load by automatically generating context (files and folders) that is based on data collected by monitoring the user and the system. CAAD [29] automatically generates context structures based on the user's work flow.

Task Tracer [13] creates task profiles based on user actions. Umea [22] monitors users' actions in order to create activity histories that it uses to determine the relevance of resources. Umea also includes personal information management tools and communication capabilities. These are, however, centralized in an standalone application rather than integrated into existing technology (operating systems). This forces users to work in two separate modes which could lead to the development of two mental models: one of the desktop system and one of the activity system.

A number of approaches aim to address the problems of knowledge workers in the digital age by integrating activity management into the desktop interface (e.g [22, 29, 33]). Project Colletta [28], Giornata [35] and the ABC system [4] closely integrate with the operating system by using a virtual desktop-like system as a structuring mechanism for activities. The latter two also consider communication and collaboration. Giornata [35] provides a contextually populated *contact palette* that can be used to share files via email and also serves as a visual cue on the amount of unread emails. In the ABC system, file sharing and real-time collaboration are supported through a pervasive framework that was designed for hospital environments [5]. Finally, Activity Explorer [27] successfully introduced an activity sharing system but limits its approach to predefined objects that are confined inside the application.

In summary, prior work has primarily focussed on either integrating individual activity/task management in the desktop interface or shared collaborative task management in separate, stand-alone applications. However, as pointed out by Moran and Zhai [26], the issues they address are dimensions of the same problem. We argue that in order to support a system that allows users to truly manage the activities they perform, the configuration of tools, coordination with collaborators and articulation (meaning distribution of awareness) of their ongoing work needs to be integrated into one desktop system. We therefore position this work at the intersection of these related works and re-analyse these three related problems informed by activity theory.

The core contribution of this paper is the design of a desktop manager that supports personal and collaborative activity-centric workflows with integrated activity-centric collaboration and interruption management tools. The novelty of this work is in how configuration, collaboration and awareness tools are included in a desktop interface; thus allowing them to become an inherent part of the integrated activity-centric workflow, rather than another tool that introduced interruptions or that needs repeated (re-)configuration.

MOTIVATION

Based on the literature review above and informed by the seven dimensions of change as described by Moran and Zhai [26], we focus on the following three problems that knowledge workers experience in working with contemporary desktop interfaces:

Pr1 : Configuration – it is cumbersome to structure the desktop and manage documents according to the wide variety of tasks one typically performs. Especially in a work setting with many parallel activities, switching between these working contexts requires a constant reconfiguration of the desktop environment thereby increasing the mental and work load of the users. The general movement from "low-level tasks to higher-level activities" (dimension 7 in [26]) requires a re-conceptualisation of the desktop interface.

Pr2 : Articulation – despite the shift "from personal to interpersonal to group to social interaction" (dimension 6 in [26]), communication and collaboration tools are separated from the tasks that use them. Most are also explicit, which means that unless users actively share information on their work context and progress, this information is not available for other users.

Pr3 : Coordination – as we are moving "from interaction with one device to interaction with information through many devices" (dimension 3 in [26]) there is still little support for sharing activities between different users and devices. The desktop work setting is intrinsically tied to the individual work context, and there is no structural support for sharing this work context with other users (which might depend on its content) or other devices.

Survey

To explore if these three problems $Pr1$, $Pr2$ and $Pr3$ are actually considered real problems by modern knowledge workers, we conducted a large scale survey. For this survey we recruited 145 participants (58% male, 42% female, average age of 33 (σ=10,06)) to reflect on these three problems through 25 representative 5-point Likert scale questions (1 = strongly disagree; 5 = strongly agree) and an open comment section (we received 112 comments). Most respondents (89,66%) described themselves as knowledge workers while 98,62% claimed to use a computer for work. Participants rated their computer literacy very high (*μ= 3.93; $\tilde{x}$= 4; σ= 0.87 on a 5-point likert scale*) and have different backgrounds (28% academic/research, 27% education, 22% IT and 23% from various other backgrounds including government, healthcare, design and media).

On average, participants owned about three devices (*μ= 2.70; σ= 1.59*) that they actively use, which is in line with the findings of Dearman and Pierce [12] and demonstrates that knowledge workers nowadays indeed use multiple devices. Interestingly, participants were very divided on whether the general management of the desktop is a problem (μ= 3.03; $\tilde{x}$= 3; σ= 1.26) and if the desktop is cluttered with to many icons and windows (μ= 2.8; $\tilde{x}$= 3; σ= 0.97). In the open question section, however, there were numerous comments that did in fact expose serious issues:

"If you don't have a well thought out workflow, your desktop/computer becomes a mess. And if you want to customize your interface, you need programming skills".

"Because of the diversity in our team (Windows/OSX users, men/women, French/English speaking, working at the office or from home,..)', we need an accessible, robust and flexible system".

Email is considered one of the most important means for collaboration (μ= 4.4; $\tilde{x}$= 5; σ= 0.83). This was emphasized by the fact that most respondents generally do not mind being interrupted from their work by email (μ= 3.5; $\tilde{x}$= 4; σ= 1.09). Interrupts via instant messaging on the other hand were not appreciated (μ= 2.4; $\tilde{x}$= 2; σ= 1.14). Participants also confirmed that they regularly share documents with colleagues (μ= 4.09; $\tilde{x}$= 4; σ= 0.90).

It was clear that sharing files and folders with contacts (μ= 3.3; $\tilde{x}$= 4; σ= 1.01) and devices (μ= 3.5; $\tilde{x}$= 4; σ= 1.03) should be much easier. Surprisingly, most respondents experience few problems in managing their open windows (μ= 2.6; $\tilde{x}$= 2; σ= 1.02). This might be explained by the small number of open windows (7) that populate their desktop (μ= 6.48; $\tilde{x}$= 5; σ= 4.29). Other comments revealed that managing child windows of a multi-window application often causes frustration and is considered to be more difficult to manage as single window applications.

In general, the survey responses demonstrated issues with the lack of separation between applications, tasks and data by the operating system. Users have to deal with the difficult task of organizing the different activities in a *workflow*[1] ($Pr1$). Unfortunately, no appropriate mechanisms are provided by current systems to automate this which leads to task fragmentation as demonstrated by [7, 11, 25]. The results of the survey also confirmed problem $Pr2$: most communication tools (such as email, instant messaging,...) require the user to provide nearly instant feedback, even if the communication received is not related to the task at hand. This is an interruption of the workflow [3, 9, 10, 36] and an important cause of fragmentation of work over time. Furthermore, users expect the availability of multiple devices [12] for similar or identical tasks, and also expect to be able to make smooth transitions between devices dedicated for work purposes ($Pr3$).

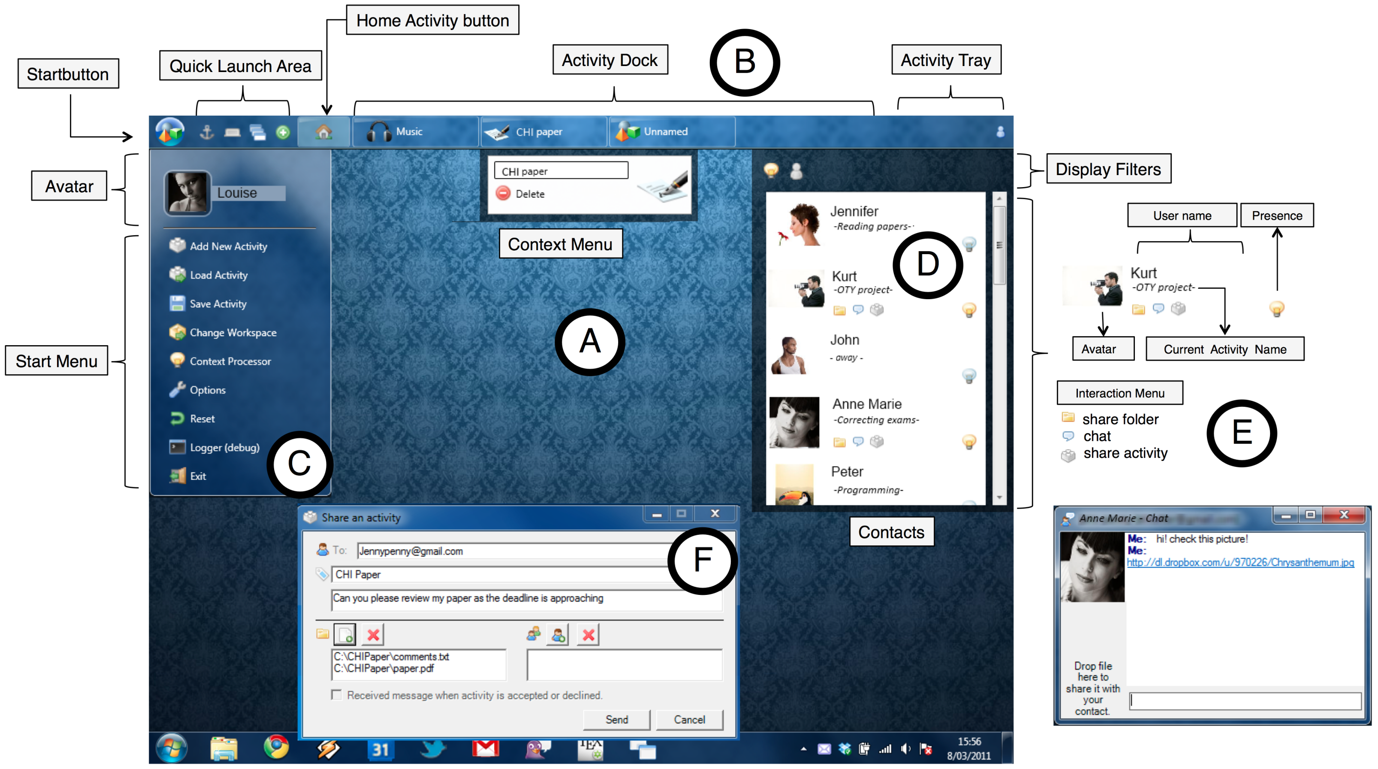
Activity Theory

To deal with these three fundamental problems, we ground our design in Activity Theory (AT) [14, 24], a descriptive psychological framework that seeks to explain human activity as a mediated and asymmetrical relation between a subject, an object and a community. An activity is engaged by a *subject* (S) that translates a need into a motive or *object* (O). This S - O relation is embedded in a *community* (C) involved in the creation of this relation. As such, the community plays an important role in the creation, development and outcome of the activity. AT suggests making these social structures part of the activity itself rather than defining them as merely external influences [24]. Finally, the S - O - C relation is mediated by *tools*, *rules* and *division of labour*. These mediators determine *how* the activity is engaged and *how* it is framed in a broader social context.

In order to make Activity Theory more concrete in context of the three problems of the contemporary desktop interface, we present three guidelines (labelled $G1$, $G2$ and $G3$ for future reference) that map directly on the problems discussed earlier. Even though these guidelines are very high level, the next section will show how these guidelines are refined into design properties that help apply Activity Theory to the design of activity-centric interactive systems.

$G1$: *Provide a shared higher level structure for organizing tasks, documents and resources*
An Activity Theory informed system should organize work using meaningful structures. Since human activity can not be reflected in a static structure but needs an evolving and dynamic structure, users should be able to *redefine* and *change* activities in the course of their work. This implies that users should be able to define and use activities according to their personal preferences. Users might use the same activities but in a different context, or just switch between activities

[1]With workflow we refer to the work practices of users.



Share an activity

To: Jennypenny@gmail.com

CHI Paper

Can you please review my paper as the deadline is approaching

C:\CHIPaper\comments.txt
C:\CHIPaper\paper.pdf

☐ Received message when activity is accepted or declined.

Send Cancel

Anne Marie - Chat

Me: hi! check this picture!
Me: http://dl.dropbox.com/u/970226/Chrysanthemum.jpg

Drop file here to share it with your contact.

Figure 2. The interface of co-Activity Manager consists of (A) a per-activity workspace, (B) an activity task bar to visualize activities to the user, (C) an activity start menu to manage activities and applications, and (D) a collaboration manager to interact and share with contacts. Each contact is visualized with an avatar, name and status field. The interaction menu (E) can be used to share a folder, chat or share an activity (F).

and save the current one to resume it later. Activity-centric software structures the workflow by connecting resources, data and activities in a semantic network specifying what resources and data are required or used by which activities. The user can manually link resources to activities or have it done automatically by the software. (Targeting $Pr1$).

*G*2: *Support collaboration and sharing*
Activity Theory includes the community explicitly and an activity-centric system should reflect this. We identify *"community"* as the participants of a social network that is making use of an activity centric system. This means that people have a shared interest in the system which should thus allow multiple people to participate in its usage and provide functionality to share (parts of) the activity of one user with other user. There must be support to distribute representations of the ongoing activities to both the user itself and other collaborators. Consistent with guideline $G1$, sharing an activity implies providing the structure of the activity as it was defined by the initiator of an activity. Collaboration and communication are part of the structure of activity and become part of an activity itself. Because communication and collaboration are an integral part of the activity, it must also be included in the representation. (Targeting $Pr2$).

*G*3: *Ensure transparent context-aware views on activities*
Most activities are not limited to a specific type of device, environment or setting which exposes the necessity of a *portable activity structure* that has the ability to transcend the individual device or user. A device-specific view on activities can be built around this portable structure. Additionally, besides the different platforms and devices, other contextual events come into play. E.g. the physical location of an individual could be used to determine the relevance of an activity. Because activity is a description that depends on space and time, it is relevant for an activity-centric system to capture both the history and context of each activity. The combination of *activity transparency* and *context awareness* provides a computational representation that maps very well on the Activity Theoretical structure of human activity. (Targeting $Pr3$)

CO-ACTIVITY MANAGER

co-Activity Manager (cAM) (Figure 2) [19] is a collaborative activity-centric desktop interface that (i) provides an activity workspace that supports ad hoc configuration of the active desktop working context, (ii) includes an activity sharing mechanism that allows for the distribution of an activity workspace, and (iii) provides built-in tool support for activity-centric collaboration. cAM deals with project fragmentation as well as communication interruptions by providing users with an activity-centric interface that allows them to (re-)organize their documents, applications and files as well as their communication and collaboration with other participants in activities.

Activity-Centric Design Properties

Activity-centric computing has proven to be a useful computing paradigm [23, 35], but remains difficult to translate into concrete software features. We therefore refine our three guidelines into six design properties which we used to inform the design of co-Activity Manager.

$D1$: *Activity-Oriented Workspace*: the design of interactive systems for knowledge workers should be focused on supporting activities. Because activities are computational representations of human activity, they should be visible in the user interface. To optimize this effect, the system should integrate with existing technology. Users must be able to create, edit, manage, consume and switch between existing computational representations of activity (refinement of $G1$).

$D2$: *Activity Sharing*: because activities are the central focus of this paradigm, activities must also (like files) be interchangeable with other users and devices. This will allow users to externalize (parts of) their activities to other contexts. Activities must therefore be platform and device independent as well as interchangeable. Activity sharing is thus not limited to the interchange between users, but also between devices (refinement of $G2$ and $G3$).

$D3$: *Activity-Centric Communication and Collaboration*: instant messaging, file sharing and other community related actions should be embedded into the activity structure to minimize "out-of-context" interruptions: interruptions generated in the context of another activity that interfere with the workflow on the current activity (refinement of $G2$).

$D4$: *Activity-Centric Presence*: as users will be able to choose their workflow as well as collaborators in the context of each activity, an advanced presence system that allows users to define availability according to the current activity is required (refinement of $G2$).

$D5$: *Activity-Centric Cloud Storage*: because activities must be interchangeable between devices and platforms and typically need to be persistent over longer periods of time, they have to be transparently stored in the cloud instead of on the local device (refinement of $G3$).

$D6$: *Activity-Oriented Context Recognition*: an Activity Theory-informed system should not only focus on providing mechanisms to create and use activities, but also to recognize changing context, such as changes in location (refinement of $G3$).

Activity-Oriented Workspace

'Activities' are implemented in co-Activity Manager as a data model that includes all resources, contacts and other (meta) information relevant to the ongoing desktop work context in an effort to reflect a physical task or activity a user is performing. The system thus provides a first class object that aligns the computational representation of data with the intention of use; the task or activity of the user.

co-Activity Manager extends the Windows 7 desktop interface with an activity-centric workspace. For each activity, a separate virtual desktop (Figure 3), that confines the working context defined by the activity, is constructed. We label this augmented virtual desktop as an *activity workspace* (on Figure 2 A). The scope of all opened windows, files and documents is limited to the activity workspace related to that activity (although windows can be transferred between or duplicated over desktops). When the user switches between activities, the workspace and Windows task manager is repopulated with windows that are related to that activity. Each activity workspace is equipped with a custom desktop folder that contains the data related to that activity. Users can simply *pile* their files and documents on the desktop per activity rather than using the hierarchical structure of the inherent file system (though this is still possible).

In order to present activities to the user, we designed an activity taskbar (Figure 2 B) that is used to manage and work on activities. By clicking the 'add' action button or by using the start menu (on Figure 2 C) the user can create a new activity. Newly created activities are by default *anonymous* and given a *default* name and icon. The user can choose to keep the activity anonymous or configure it for more persistence. The name, icon and other information can be changed at any time by simply launching the context menu. For each newly created (or loaded) activity, the system adds a new activity button to the dock. By pressing an activity button, the associated activity workspace is loaded causing the desktop icons, windows and the build-in collaboration manager (Figure 2 D) to update.

All activities that are located on the activity dock of the taskbar are part of the same *activity workbench*. Users can create and delete workbenches or change existing workbenches to configure them according to their personal work preferences. This can happen both before and after sharing an activity with others. Activity workbenches are introduced as a feature to deal with both activity clutter or overpopulation of the activity taskbar as well as provide a meta structure to manage activities.

For the design of this activity taskbar, we mimicked the standard Windows 7 taskbar but redesigned it to be suited for activity management (Figure 2 B) as demonstrated in prior work [4, 33]). By exploiting user's *familiarity* with the taskbar, we expected to get a higher level of user acceptance since activity management can be operated similar to how one manages applications. Documents, windows and applications are no longer loosely coupled elements that float around the desktop but are embedded in an activity. When one switches activities, all documents, windows and applications will also change accordingly.

Activity-Centric Cloud Services

To overcome the fragmentation that is caused by the wide variety of devices being used [12], we make use of cloud storage. This means that files, documents and activities are stored online but are transparently accessible through the desktop interface. This approach is an effort to integrate the Personal Information Cloud [26] into cAM. cAM allows users to save and load activities, and their containing files and configuration from the cloud storage. cAM automatically builds an XML file of the activity and saves this along with all resources. This file can afterwards be imported on another devices that runs cAM or another piece of software that supports this format. In the current version, cAM integrates with the

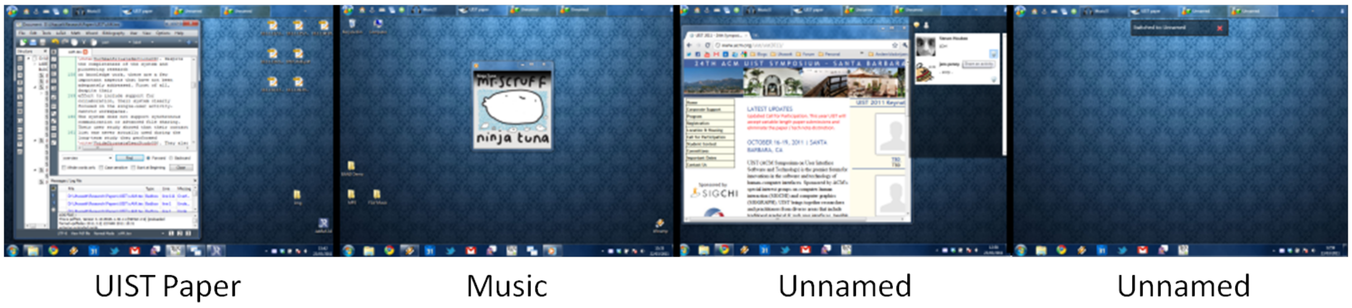


Figure 3. Each activity has its own activity workspace.

Dropbox API [2] to support cloud storage, but because of the use of the XML standard, any type of storage can be used.

Activity-Centric Communication and Collaboration

The activity management system of co-Activity Manager does not only consider windows, applications and documents but also includes tools for communication and collaboration. Since most local task fragmentation is caused by interruptions of instant messengers and other communication tools, we argue that the workflow can not be restructured without explicitly including built-in communication and collaboration. By including both communication and collaboration tools in the structure of an activity, we thus aim to decrease work fragmentation.

cAM includes a collaboration manager (Figure 2 D) that can be used to interact with collaborators. First of all, we support standard *chat* messages. The chat window is equipped with an automatic sharing system, that allows users to drag and drop documents they want to share on top of the chat windows. These files are then automatically uploaded to the cloud storage and shared through the chat window. Secondly, the user can define a *shared folder* for each contact per activity. All these folders and their content are stored in the cloud storage and can be used as a persistent sharing mechanism or to share large amounts of data. Finally, users can also *share activities* (on Figure 2 F).

Because one of our goals was to create a *usable* and *scalable* system, we use the Gmail infrastructure for communication and collaboration. Since their email system is free, already widespread, very robust and also provides a distributed contact list, it was best suited for integration into co-Activity Manager. Additionally, Gmail also provides XMPP [32] support for real-time communication, which allows us to develop custom XMPP extensions for other purposes (including activity sharing). Summarized, cAM integrates messaging, file sharing and activity sharing into the desktop interface.

Activity-Centric Presence

As the collaboration manager (on Figure 2 D) is included in the design of the activity workspace, it can be populated for each activity. Users can choose which contacts they finds relevant for each activity. By adding these contacts to the collaboration list of the specific activity, they define a group of contacts who are allowed to communicate with them and who are made visible for the users themselves. By default, all collaborators are added to the list, but not added to the activity.

The collaboration manager also provides association and online/offline filters (Figure 2 E) to customize the list to personal preferences for each activity. These filters remove or show offline/online or associated/not associated contacts. When users switch between activities, the collaboration list is updated and the contacts will see the user appear as offline or online depending on if they are added to the loaded activity or not. If the user is online, the name of the activity the user is currently engaging will be distributed to all collaborators.

This activity-centric presence system allows users to not only control their workflow but also their communication flow. We believe that by allowing users to control who is allowed to interrupt them, they can also better control the entire workflow, therefore decreasing task and work fragmentation. In order to add or remove a user, the user simply clicks the light bulb (on Figure 2 E) button that is located on the interaction menu of each contact. When the light is on, the contact is associated with the activity and thus allowed to interact with the user. In this case, the user will be reported as available in the contact list. When the light is off, the contact is not associated with the activity. In this case, the user will be reported as unavailable in the contact list (but still online) and cannot interrupt the current activity by sending a direct message. In order to distribute these presence changes to all contacts, we use a custom XMPP extension that carries the activity-based presence information to all collaborators.

Activity Sharing

Since the entire approach is focused on activities and its components, we also argue that this structure must be interchangeable between users and devices. This implies that the user is not only defining their own ad hoc workflow, but can also suggest the workflow of their collaborators or other devices. This vision also embraces an entire activity life cycle rather than perceiving activities as a individual local organizational structure.

In the collaboration manager (Figure 2 D) the user can launch an activity delegator window (on Figure 2 F), that can be used to describe the activity the user wants to share. These descriptions include a name, a textual description of the goals and motivation of the activity, a set of resources (e.g. local documents, folders, applications), relevant contacts and the previously recorded history as configured in the local activity

[2]https://www.dropbox.com/developers

workspace. After resources are stored in the cloud, the system sends the description of the activity to the selected contact.

The receiving user is notified that there is a new activity; accepting it will cause the system to construct a new local activity, based on the description in the XML file that was received. The name, icon, description and other activity information is used to define the local activity; the related resources are downloaded from the cloud to the workspace of the activity and the collaboration manager is populated with relevant contacts. The activity is grabbed from the cloud and locally deployed. This mechanism thus allows users to preconfigure an activity and share this configuration with other collaborators, who in turn can deploy and customize it based on their personal preferences.

Activity-Oriented Context Recognition

Although we want to emphasize the importance of context recognition for a holistic approach to activity representation, this has not been the main focus of co-Activity Manager. As a proof of concept, we developed a workbench switcher that dynamically changes activity workbench depending on the connected wireless network SSID. This implies that the user can couple a set of related activities to a specific physical environment. The contextual workbench switcher was included to deal with activity clutter. We did not include any additional support for context awareness in this prototype, but included an API that can be used by external context-awareness or monitoring systems such as Subtle [16] and PersonalVibe [8] in order to update or change the interface based on external events.

FIELD DEPLOYMENT

co-Activity Manager was deployed for a two week period in a five-person multidisciplinary software development team – consisting of software engineers, a graphic designer and historian. The team specializes in developing interactive setups for cultural heritage sites such as museums or tourist attractions. We selected this software development team for four reasons: (i) they spend several hours a day using a desktop interface; (ii) they all tend to collaborate in (partially) overlapping subgroups (see Figure 1); (iii) due to its multidisciplinary character, the team is composed of people with varying computer expertise; and (iv) they each currently tend to structure their workflow in different ways, which allowed us to discuss the value and potential of activities as a workflow structuring mechanism.

The primary goal of this evaluation was not to assess the *usability* of co-Activity Manager but rather to explore the *feasibility* of a collaborative *activity-centric desktop interface* for knowledge workers through a case study. Because this cannot be tested in a lab environment [34], we deployed the system on the primary computer of all participants and asked them to use it for their daily work activities during office hours. Although this study provides valuable insights in the multi-user usage patterns and immediate issues arising from using this type of system in a real world setting, additional longitudinal studies are needed to confirm the usability for a broader spectrum of users.

Experimental Setup

The team consisted of five people in total (3 male, 2 female; mean age = 31): three software engineers (P1, P2 and P3), a graphic designer (P4) and a historian (P5). Before deploying the system, we conducted a short pre-study interview in which we discussed with participants how they structure their work on the desktop and with which other team members they regularly collaborate. Before the deployment, participants received a short demonstration of the features of co-Activity Manager.

For the two-week deployment, participants were instructed to use the system in the course of their day-to-day work. During this period, participants were observed and interviewed regularly to detect potential problems and discover emerging behaviour. At the end of this period, participants were asked to complete a short questionnaire which was used as a basis for a semi-structural interview in which we discussed the user's experiences and opinions of the usefulness of the concepts of co-Activity Manager.

Results

During the pre-study interviews, all participants reported doing both *individual* and *highly collaborative* work. Moreover, we also observed that participants used quite diverse ways of structuring their workflow, including the way they stored important documents, organized their windows and managed communication and collaboration with others. We were interested to see how these differences would affect their appreciation of co-Activity Manager, and if co-Activity Manager would be flexible enough to cope with these different ways of working.

Table 1 lists the results of the 5-point likert scale survey which ranges from "not at all useful" (1) to "very useful" (5).

Questions	*Min*	*Q1*	$\tilde{x}$	*Q3*	*Max*	*Iqr*
Usefulness cAM	3	3.75	4	4	4	0.25
Activity-centric concept	1	3.25	4	4.25	5	1
Sharing Activity Workspace	1	3.25	4	4	4	0,75
Activity-centric collaboration	3	3.75	4	4	4	0.25
Activity Cloud support	3	3.75	4	4	4	0.25

Table 1. The result of the 5-point likert survey that was used as a basis for the interview. The table shows an overview of the minimum, maximum, median ($\tilde{x}$) and the inter quartile range (iqr).

Activity Workspace

When asked about the design of co-Activity Manager, participants reported that they liked the overall design of the system as it helped them focus better on the active working context as well as find the information inside this working context faster ($\tilde{x}$= 4; *iqr*= 0.25). Participants also appreciated the general notion of *activities* ($\tilde{x}$= 4; *iqr*= 1). However, in strong contrast to his colleagues one participant (P3) did not like the concept of activity at all as he argued that it would increase complexity rather than decrease it. Surprisingly, during our observations he in fact did use activities (to show or hide a remote desktop connection).

The graphic designer (P4) also liked the idea of activities but did not like the implementation because it did not match with his Mac OS X workspace. P1 felt that activities were very useful as they allowed him to structure his workflow based on the parallel ongoing projects. He also felt that using activities helped him focus better on his work because he felt less distracted. P2 used activities in a similar manner as P1 as she created an activity for each project. The historian (P5) used activities in a rather unexpected way. At the beginning of each workday, she created a set of activities which mapped directly to tasks she was planning to do that day. During the day she would work through the list of activities one by one, keeping the finished ones as a reference. Although we did not anticipate participants would use activities for managing a to-do list, this approach worked very well for her.

Activity Sharing

Activity sharing ($\tilde{x}$= 4; *iqr*= 0.75) did not occur frequently but was rather used as an initiation process for a new long term collaborative project. During the observation, we noticed that when P1 received information (documents, images and other resources) for a new project that also involved another colleague (P2), he created a new activity that contained all this information and shared it with P2, who then accepted the activity and deployed it on her own machine as a local activity workspace. They both liked that instead of having to copy all the documents and resources manually, they could simply *share an entire context* which is automatically deployed on a new desktop. We also noticed that after receiving and deploying the activity, P1 and P2 would both customize (i) the activity itself as well as (ii) the activity workspace to their specific role. During the interview, P2 explained:

> *"It is much easier to just receive an entire activity and then customize it, than to find and collect all information separately".*

Both users reorganized the activity workspace based on their own preferences but more importantly deleted files, folders and contacts that were not relevant for their specific part in the activity.

Activity-Centric Collaboration

The integrated collaboration features, such as folder sharing and per-activity contact lists were used much more than the activity sharing mechanism. Most participants argued that the per-activity communication filter was very useful ($\tilde{x}$= 4; *iqr*= 0.25) as a tool to channel communication streams into the active working context. It was especially considered important for people that do a lot of multitasking or that are working at several active projects on the same time. The historian (P5) found this way of working very valuable as she argued that now all relevant information as well as contacts were visible and managed inside one desktop workspace. During the deployment it also quickly became clear that the integrated instant messenger would not be used very much. Because all participants in this study were physically collocated there was no active need to have a chat communication channel with all participants.

P1, e.g., used the per-activity contact list as a starting point for file and activity sharing but confirmed our observation that the IM functionality was a *second communication back channel* since most of the discussions were done face to face in the room. He also requested to add functionality that would allow certain contacts to be automatically added to all activities. In his case, he wanted to be available to his wife at all times regardless of what working context he was in. Finally, the historian (P5) also saw the potential in per-activity IM functionality, but she argued that the main advantage of this would be in the case when external collaborators that were not physically collocated would also use cAM.

During the study, a number of privacy issues arose from the mixture of private and work-related communication. As co-Activity Manager integrates with the Google Mail infrastructure, the activity-centric presence of all users was automatically distributed to all contacts in their Google Talk contact list. This side-effect of the chosen technology raised some serious privacy, reliability and confidentiality issues. A friend of P2, e.g., confronted her with a still undisclosed name of a project she was working on. Because she named one of her activities according to this new project, this name was distributed to all Google Talk contacts.

Automatically distributing people's current work activity was perceived to be very useful but also intrusive. In general people liked having an overview of who was working on what project since this knowledge would spark ad hoc meetings or collaborative work. However, it also had some disadvantages. P3, e.g., renamed one of his activities so that it would seem as work-related, even though in fact the activity only contained personal content such as a chat window to his wife and his music player:

> *"I didn't like that other people could see I was in an activity that was not strictly related to my work. In the end I renamed the activity, but you should have the ability to determine for each activity if you want to share the content".*

He felt that this information was irrelevant for his colleagues and he therefore did not want his colleagues to be informed of this. Finally, the interviews exposed that the per-activity contact list would, at times, cause inconvenient situations. Participants sometimes switched activities during a conversation which resulted in muting the other contact (if they did not happen to be part of that other activity). The muted contact would then have no way of leaving a message or response. P2 argued that asynchronous messaging should therefore be integrated into the system (as also proposed earlier by Voida et al. [35]), but P1 disagreed with this as he argued that email could be easily launched as part of the ongoing activity.

Activity Cloud

All participants in our study were actively using two to four devices so most of them valued the idea ($\tilde{x}$= 4; *iqr*= 0.25) of saving their activities in a cloud store as it would enable them to load and save activities on different devices that was equipped with a co-Activity Manager. P1 liked cloud support not only because it could be used to distribute activities over multiple devices, but also because the cloud storage mechanism allowed him to backup contextually meaningful structures rather than just a set of files.

DISCUSSION: LESSONS LEARNED

The central focus of co-Activity Manager is to provide an integrated activity-centric solution for the (i) configuration, (ii) articulation and (iii) coordination problem that occurs in modern knowledge work. In this section, we discuss the lessons learned from the implementation and deployment of the study as well as future directions.

Our study demonstrates that a relatively lightweight and open activity workspace allows for flexible activity management in different work practices. All participants in the study used activities in different ways, e.g., to organize projects, as a to-do list or simply as an extra desktop. And both the *duration* as well as the *scope* of the activities greatly differed between users.

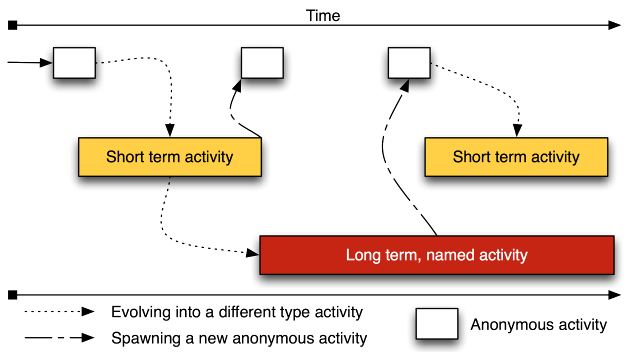

Time
Short term activity
Short term activity
Long term, named activity
Evolving into a different type activity
Spawning a new anonymous activity
Anonymous activity

Figure 4. Activity Lifecycle – activities always start as empty anonymous ad hoc activities before disappearing or evolving into short or long-term defined activities. From within these defined activities, new ad hoc activities would emerge, that in turn would disappear or evolve.

Activities created and used during the study, can be categorized as either (i) long-term defined activities, (ii) short-term defined activities and (iii) ad hoc anonymous activities. The long term activities were given a proper name, icon and description, and were mostly used for long term projects or permanent activities (such as personal activities containing a music player and open email client). The short-term activities were also marked with an icon and named but had a shorter life span. The to-do approach of one participant is an example of using short-term activities, which were created in the morning and deleted at the end of the day. The difference between the different types of activities is not necessary the duration but rather the intentionality of the activity; the reason why the activity was created and used. Finally, most users also used ad hoc anonymous activities. Because creating a new activity is easy (one button press and no a priori configuration requirement), it was used for basic operations that required a clean desk or separate working context (e.g. writing a quick email or copying files from one folder to another).

The typical lifecycle of an activity is illustrated in Figure 4. Each activity starts as an empty ad hoc anonymous activity but evolves over time to a short or long-term activity with a proper name, definition and visualization depending on the content or the ongoing work context. From within emerging activities new ad hoc activities would be spawned that either died very quickly or evolved into another short or long-term defined activity. By allowing users to store the configured activity into a cloud store, they become usable beyond the individual device.

A key goal of co-Activity Manager was to include communication and articulation channels into the activity abstraction. By providing a collaboration manager that is customizable for each activity, we provide a mechanism for users to tunnel communication into the appropriate working context. In our implementation, we focus specifically on frequently used collaboration and communication tools.

Activity-centric collaboration emerged in two distinct phases: (i) a *sharing* phase in which a short- or long term activity would be prepared, shared and deployed by collaborations in order to start the (ii) *coordination* phase in which other users would use the per-activity collaboration manager to actually consume the collaboration inside a workspace. This process allowed users to use the configurations of their collaborator as a starting point for the configuration process of their own activity as part of the collaboration. The effect of this is that users can essentially template a work context and share it as a joint starting point in a collaborative setup for a team. Participants themselves proposed the notion of *workflow* to define this process and argued that this mechanism could even be used in a more restricted way.

As new activities would emerge for a local user, new collaborations would be spawned from within existing activities by using the collaboration manager. This per-activity collaboration manager, however, had several other roles. First of all, it was used as a default starting point for all types of communication and collaboration. Despite the fact that our prototype had technical limitations (e.g. we only support one IM protocol), it was used as intended and participants had many suggestions for improvements. During interviews, however, it became clear that for users to accept this type of system, it needs to *integrate* with the tools (protocols) they know and use now. Second, the manager also functioned as an awareness tool as it distributed the working context of all collaborators, thereby sparking face-to-face discussions and collaborations.

During our study, a number of *privacy* and *confidentiality* problems arose because of the automatic distribution of information of the active work context. Some of these problems were related to the technological implementation but further investigation exposed a more complex problem at the intersection of organizational policies and personal preferences. The team that was part of our user study was allowed by its employer to use email and instant messaging for both work and personal purposes. This, of course, greatly complicates the process of distributing information as private and public space is mingled into one interface. Potential solutions to this could include a higher level of control (e.g. Access Control Lists) over the distribution process (which would greatly complicate the process) or organizational policies that define the scope of the distribution on an infrastructural or protocol level.

CONCLUSION

In this paper we introduced co-Activity Manager (cAM), an activity-centric extension of the Window 7 interface that aims to deal with the problems and limitations of the desktop interface in context of collaborative knowledge work. cAM includes (i) an activity-centric workspace to minimize the configuration load, (ii) an activity sharing mechanism that can be used to distribute collaborative activities, and (iii) a per-activity collaboration manager that helps users tunnel communication channel and setup collaborations. We reported on a 14 days deployment in a multi-disciplinary software development team. Our study showed that the activity workspace is flexible enough to accommodate different individual and collaborative work practices and that activity-centric collaboration is a two-phase process consisting of an activity sharing and per-activity coordination phase.

ACKNOWLEDGEMENTS

This work was supported by the EU Marie Curie Network iCareNet under grant number 264738.

REFERENCES

1. E. Adar, D. Karger, and L. A. Stein. Haystack: per-user information environments. In *Proc. of CIKM '99*, pages 413–422. ACM.
2. A. Agarawala and R. Balakrishnan. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *Proc. of CHI '06*, pages 1283–1292. ACM.
3. B. P. Bailey, J. A. Konstan, and J. V. Carlis. The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In *Proc. of INTERACT '01*, pages 593–601. IOS Press.
4. J. Bardram, J. Bunde-Pedersen, and M. Soegaard. Support for activity-based computing in a personal computing operating system. In *Proc. of CHI '06*, pages 211–220. ACM.
5. J. E. Bardram. Activity-based computing for medical work in hospitals. *ACM Trans. Comput.-Hum. Interact.*, 16:10:1–10:36, June 2009.
6. V. Bellotti, N. Ducheneaut, M. Howard, and I. Smith. Taking email to task: the design and evaluation of a task management centered email tool. In *Proc. of CHI '03*, pages 345–352. ACM.
7. O. Bergman, R. Beyth-Marom, and R. Nachmias. The project fragmentation problem in personal information management. In *Proc. of CHI '06*, pages 271–274. ACM.
8. A. B. Brush, B. R. Meyers, D. S. Tan, and M. Czerwinski. Understanding memory triggers for task tracking. In *Proc. of CHI '07*, pages 947–950. ACM.
9. A. F. Cameron and J. Webster. Unintended consequences of emerging communication technologies: Instant Messaging in the workplace. *Computers in Human Behavior*, 21(1):85–103, Jan. 2005.
10. E. B. Cutrell, M. Czerwinski, and E. Horvitz. Effects of instant messaging interruptions on computing tasks. In *In CHI '00 EA*, pages 99–100. ACM.
11. M. Czerwinski, E. Horvitz, and S. Wilhite. A diary study of task switching and interruptions. In *Proc. of CHI '04*, pages 175–182. ACM.
12. D. Dearman and J. S. Pierce. It's on my other computer!: computing with multiple devices. In *Proc. on Human factors '08*, CHI '08, pages 767–776. ACM.
13. A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. Tasktracer: a desktop environment to support multi-tasking knowledge workers. In *Proc. of IUI '05*, pages 75–82. ACM.
14. Y. Engestrom. *Learning by expanding: an activity-theoretical approach to developmental research*. Orienta-Konsultit Oy., 1987.
15. S. Fertig, E. Freeman, and D. Gelernter. Lifestreams: an alternative to the desktop metaphor. In *Proc. of CHI '96*, pages 410–411.
16. J. Fogarty and S. E. Hudson. Toolkit support for developing and deploying sensor-based statistical models of human situations. In *Proc. of CHI '07*, pages 135–144.
17. D. A. Henderson, Jr. and S. Card. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Trans. Graph.*, 5(3):211–243, 1986.
18. E. Horvitz and J. Apacible. Learning and reasoning about interruption. In *Proc. of ECMI '03*, pages 20–27. ACM.
19. S. Houben, J. Vermeulen, K. Luyten, and K. Coninx. Co-activity manager: integrating activity-based collaboration into the desktop interface. In *Proc. AVI 2012*, pages 398–401. ACM.
20. D. R. Hutchings and J. Stasko. Quickspace: new operations for the desktop metaphor. In *Proc. of CHI '02*, pages 802–803. ACM.
21. J. Jin and L. A. Dabbish. Self-interruption on the computer: a typology of discretionary task interleaving. In *Proc. of CHI '09*, pages 1799–1808. ACM.
22. V. Kaptelinin. Umea: translating interaction histories into project contexts. In *Proc. of CHI '03*, pages 353–360. ACM.
23. V. Kaptelinin and M. Czerwinski, editors. *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments*, volume 1 of *MIT Press Books*. The MIT Press, 2007.
24. V. Kaptelinin and B. A. Nardi. *Acting with Technology: Activity Theory and Interaction Design (Acting with Technology)*. The MIT Press, 2006.
25. G. Mark, V. M. Gonzalez, and J. Harris. No task left behind?: examining the nature of fragmented work. In *Proc. of CHI '05*, pages 321–330. ACM.
26. T. P. Moran and S. Zhai. *Beyond the desktop in seven dimensions*, chapter 11, pages 335–354. The MIT Press, 2007.
27. M. J. Muller, W. Geyer, B. Brownholtz, E. Wilcox, and D. R. Millen. One-hundred days in an activity-centric collaboration environment based on shared objects. In *Proc. of CHI '04*, pages 375–382. ACM.
28. G. Oleksik, M. L. Wilson, C. Tashman, E. Mendes Rodrigues, G. Kazai, G. Smyth, N. Milic-Frayling, and R. Jones. Lightweight tagging expands information and activity management practices. In *Proc. of CHI '09*, pages 279–288. ACM.
29. T. Rattenbury and J. Canny. Caad: an automatic task support system. In *Proc. of CHI '07*, pages 687–696. ACM.
30. M. Ringel. When one isn't enough: an analysis of virtual desktop usage strategies and their implications for design. In *Proc. of CHI '03*, pages 762–763. ACM.
31. G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel, and V. Gorokhovsky. The task gallery: a 3d window manager. In *Proc. of CHI '00*, pages 494–501. ACM.
32. P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. RFC 3921 (Proposed Standard), Oct. 2004. Obsoleted by RFC 6121.
33. G. Smith, P. Baudisch, G. Robertson, M. Czerwinski, B. Meyers, D. Robbins, and D. Andrews. Groupbar: The taskbar evolved. In *Proc. of OZCHI '03*, pages 34–43.
34. S. Voida and E. D. Mynatt. "it feels better than filing": Everyday work experiences in an activity-based computing system. In *Proc. of CHI 09*, pages 259–268. ACM Press.
35. S. Voida, E. D. Mynatt, and W. K. Edwards. Re-framing the desktop interface around the activities of knowledge work. In *Proc. of UIST '08*, pages 211–220. ACM.
36. F. R. Zijlstra, R. A. Roe, A. B. Leonora, and I. Krediet. Temporal factors in mental work: Effects of interrupted activities. Open Access publications from Maastricht University urn:nbn:nl:ui:27-18293, Maastricht University, 1999.