Managing Activities in Time and Space in an Activity-Based Computing System

Steven Jeuris

Supervisors:

Prof. Dr. Peter Werkhoven	Utrecht University
Dr. Robbert-Jan Beun	Utrecht University
Ingrid van Zaanen, Project Manager	Logica
Friso van Waveren, MS Software Engineer	Logica

Master's thesis in Game and Media Technology
Thesis number: ICA-3322092
Utrecht University
Academic year 2011-2012

Acknowledgements

A number of people have made this thesis possible. First and foremost I would like to express my gratitude to my supervisors, in particular Prof. Dr. Peter Werkhoven from the University of Utrecht who allowed me to work on this topic. Being able to work on a self-proposed subject helped me in staying highly motivated throughout the project. Thank you for taking the time throughout your busy schedule and providing me with valuable feedback. Likewise I would like to thank everyone at Logica and Logica Working Tomorrow where I did my internship to work on this thesis. In particular Ingrid van Zaanen and Friso van Waveren who have made themselves available whenever I needed help.

I'd also like to explicitly thank Stefan Calle, who has allowed me to work part-time at AIM Productions while studying at the University of Utrecht. Without the great amounts of offered flexibility I would not have been able to combine work and studies. I am indebted to all of my colleagues at AIM for the pleasant working environment and provided understanding during the more stressful times. All of you have contributed greatly to my studies at one point or another.

It has been an honor for me to be able to continue work on open research questions as specified by Prof. Dr. Jakob E. Bardram from the IT University of Copenhagen. Your existing work helped me in deciding where to place the focus in this dissertation. Thank you for bringing me into contact with MSc Steven Houben, whose thesis was a great inspiration to me, who was regularly able to answer questions I had, and whose feedback on initial drafts of my thesis helped in shaping my conclusions.

This thesis would not have been possible without all the people who tested the prototype interface and gave extensive feedback. I owe you my gratitude for taking up so much of your time to help me out. Your positive feedback has been a great inspiration while writing the final chapters of this thesis.

Last but not least, I would like to thank my family and friends for the continued support even through most stressful times. I know I have often been preoccupied, and haven't always been as pleasant to be around with, for

which I apologize. In particular I would like to thank my mother for helping me in finding suitable participants for the user study, which helped me in obtaining a diverse set of testers. Bouncing ideas off of Sanne Alsters has been a great inspiration for the eventual layout of the user interface, and her creative input is more than appreciated. Lastly, I would like to acknowledge the help I have gotten from the on line Stack Exchange community, in particular some acquaintances over at Cognitive Sciences. It continues to amaze me how crowdsourcing is such an effective form of information sharing and learning.

Abstract

While most designs of information technology are based on the notion of supporting distinct tasks, research has shown that "people organize their work in terms of much larger and thematically connected units of work". [18] Current systems are notoriously bad at supporting this. Switching between these units of work (activities) requires the user to find the relevant applications and documents associated with them each time. After closing the activity it can't be restored, and working on multiple activities in parallel causes information overload.

An Activity-Based Computing (ABC) system supplements the prevalent data- and application-oriented computing paradigm with technologies for handling multiple, parallel and mobile (cross-device) work activities. [5] Several implementations have demonstrated the advantages of such a system, but some open issues remain. [6] Organizing a large number of activities is troublesome. The problem of managing a lot of applications shifted to managing activities.

Since an agenda is a known useful tool to plan activities, applying this metaphor to an ABC system seems logical. Activities can be organized in time to indicate when work was done on them, or is planned. Space can be used to indicate how activities relate to each other.

This project demonstrates a system to manage activities in time and space as part of an ABC system on Windows 7, addressing some of the known issues when working on multiple activities. The design is evaluated by performing a user study.

Contents

Chapter 1

Introduction

> *Applications: what a terrible term. What a terrible concept. Applications have little to do with the tasks that people are attempting to accomplish. Look. We don't do word processing; we write letters, or memos, or reports, or notes to ourselves.*
>
> – Don Norman [32]

Computers are inherently application and document oriented. Existing research suggests that people organize their work in terms of much larger and thematically connected units of work. Up to 45 minutes a day are spent on managing these units of work. [18] Mainstream graphical user interfaces (UIs) as we still use today don't support the user in managing or sharing these "activities" that well. Switching between activities requires the user to find the relevant applications and documents each time. Sharing an activity requires manually selecting documents and sending them to collaborators. Activity theory [27] indicates a user interface which focuses on semantically meaningful activities instead, might correspond better to the mental mapping of knowledge workers and offload some of the overhead present in current UIs. To address these issues several Activity-Based Computing (ABC) systems have been created and were well received. [5, 39] However, only small aspects of such a system are presently available in some of the mainstream systems, and several aspects remain unresolved.

1.1 Relevance

As a heavy computer user I experience several shortcomings of UIs firsthand. Research suggests that switching between task contexts should be supported by information technology. [27, 18, 12] A lot of knowledge workers use information systems on a daily basis, making this research highly relevant to

a wide variety of users. A well designed ABC system could possibly be used as a basis for, or even replace some business specific software packages which manage small activity-oriented data sets.

This research continues on some open issues in ABC system research as mentioned by Bardram. [6]

> *Tools and methods for handling, linking together, and navigating in a complex web of activities are needed.*

Current ABC systems don't scale that well. A user can accumulate a huge amount of activities, of which some get out-dated after a while. A similar problem arises as the one we set out to resolve in the first place. Allowing the user to organize and navigate their activities is critical for the system to be usable during longer time periods.

> *Activity Life Cycle: one of the recurrent observations was that one activity merged into another – there were no clear demarcations of the ending of one activity.*

Although it makes sense to organize work in a set of activities, users don't always know upfront when a new activity will arise from another. Only after a while they might notice the need to separate part of the resources they are working on into a different activity. On the other hand, users might want to merge two activities together as the resources of both should be combined. Activities should be light-weight, easy to create and modify.

> *How can you support one user in handing over an activity to the other participants in the same activity?*

Besides forming artifacts while working on an activity, users also form a mental model of the activity in their working memory. When handing over an activity to another user, this mental model is lost. Cues should be provided on what the original user was doing so the new user can interpret the artifacts more easily. A similar problem may occur when users revisit an old activity they created on which they haven't worked for a while.

1.2 Research questions

A wide range of conceptual requirements for an ABC system have been formulated in existing research, which will be discussed in Section 2 and later summarized and supplemented in Section 4.1. However, the scope of this research is limited to the following research questions. Only those conceptual

requirements relating to them will be considered during the design, implementation and evaluation of the user interface.

> *Main question: Can organizing activities in time and space simplify managing them and reduce information overload?*

ABC systems show a lot of promise, but some open issues remain. [6] Organizing a large number of activities is troublesome. The problem of managing a lot of applications and documents shifted to managing activities. Since an agenda is a known useful tool to organize activities, applying this metaphor to an ABC system seems logical. Taking advantage of recent UI technologies, 3D space can be used to lay out and interact with interface elements. By organizing activities in time, users do not only have an overview of active activities and their history, but also finished ones. Space can be used to show relations between activities; which activity arised out of which, or do they share common resources. Although activity collaboration won't be a subject of this thesis, displaying the history of activities, and optionally allowing the user to add annotations, could be useful for activity sharing. Two supporting sub questions to this main question can be identified.

> *Q1: Can centralized handling of application notifications help in classifying activities and handling interruptions?*

There are no clear borders in between different activities; they often merge into each other. Where does one activity end and when to create a new one? The proposed solution embraces this light-weightiness of activities, by allowing easy splitting or merging of activities and displaying these actions – the history and emergence of the activities – in time.

Although workers may switch among tasks in a self-guided manner, a significant portion of task switching is caused by external interruptions. [12] Centralized notification handling attempts to link external interruptions to new or existing activities, further aiding the user in maintaining an overview. These interruptions (e.g. emails) can be handled in a centralized system which can immediately handle, ignore or link them to a new or existing activity.

> *Q2: How can an Activity-Based Computing system be integrated into an existing operating system?*

In order for ABC systems to be usable in existing operating systems (OS's) and measure up to the requirements of easy task switching [10] some application-wide behavior will need to be implemented. Applications need to

become aware of activities. Since current OS's don't require such behavior from existing applications, fallback methods need to be explored.

A prototype for Windows 7 will be created and evaluated by a user study, comparing the proposed solution with a traditional user interface.

Chapter 2

Existing Research

Already in the early days of Human-Computer Interaction (HCI) research Bannon et al. (1983) [4] identified the lack of "support for the kinds of problems users encounter when attempting to accomplish several different tasks in a single session". Based on their analyses they provided a number of conceptual guidelines for developing an interface which supports activity coordination. Card et al. [10] refined these requirements and implemented them in a virtual-workspace interface, Rooms. (Section 2.1.1) Throughout the years, several similar approaches have been researched, and additional requirements were added and evaluated. Recent examples include the Activity-Based Computing system (Section 2.1.2) by Bardram et al. [5] and Giornata (Section 2.1.3) by Voida et al. [39].

Similar requirements are recognized in several papers but often different terminology is used. We will stick to the terminology used within the discussed papers. In section 4.1, we will review the existing requirements and frame them within a model adopted from activity theory. Following this model allows us to identify new requirements where previously none were defined.

2.1 Activity-Based Computing Systems

In his book "The Invisible Computer: Why Good Products Can Fail" [32], Don Norman, one of the founders in the field of human-computer interaction (HCI), describes an approach to computing called "Activity-Based Computing" which was proposed by a "hardy band of souls" at Apple Computer as a paradigm which would fit the lives of their customers better than the traditional application model. Unfortunately it never did make it to product, mainly since it was a disruptive technology description offered to an indus-

try that wasn't interested at that time. ABC was strongly and explicitely inspired by activity theory.

Bardram et al. [5] identified an ABC system as a system which "supplements the prevalent data- and application-oriented computing paradigm with technologies for handling multiple, parallel and mobile work activities". "In activity-based computing, the basic computational unit is no longer the file (e.g. a document) or the application (e.g. MS Word) but the activity of a user". They named their system the same way, but in this thesis we will use it as a general term to refer to systems which incorporate similar concepts.

In this section, an overview of conceptual requirements for an ABC system brought forward by related research is given, along with a short description of the resulting user interface.

2.1.1 Rooms

Card et al. (1986) [10] continued on the work of Bannon et al. [4] and identified several desirable properties of an interface which supports multiple activities, as summarized in Table 2.1. Their focus was on solving the problem of limited screen space by allowing the user to manage a set of different tasks between which could be switched. They set out to create a no-compromise design where only the desired information related to the task at hand is visible, while still allowing to quickly access all other data sources with low overhead. Different tools can be shared across several tasks and can have a different representation best suited for a given task.

Task Switching Properties	
A1	Fast task switching.
A2	Fast task resumption.
A3	Easy to re-acquire mental task context.
Information Access Properties	
B1	Access to a large amount of information.
B2	Fast access to information.
B3	Low Overhead
Interactions Among Tasks	
C1	Engaged Tools sharable among several Tasks.
C2	Collections of Engaged-Tools sharable among tasks.
C3	Task-specific presentations of shared Engaged-Tools.

Table 2.1: Desired properties as defined by Card et al.

Their approach is to provide the user with a suite of different screen-

sized workspaces called Rooms, which highly resemble what are now more commonly known as virtual desktops. Each Room contains a set of window Placements, which store which window is located where, including its size. When a Room is visible only the Placements associated with that Room are displayed. Two Placements may refer to the same window, allowing windows (and thus Tools) to be shared among Rooms, while having a different location and size in each. Rooms can lead to other Rooms, represented by a Door on the desktop. To help the user navigate, the system has an Overview that displays miniature versions of all the Rooms in the total user workspace. Placements can be copied, deleted, moved, shaped and examined at full size from this Overview. Additional features allow to move Placements from one Room to another, and to show some Placements across all Rooms. The result is the total virtual workspace is divided into multiple virtual sub-workspaces, where separate tasks can be performed. The entire system can be restored so that users never have to reinitialize an entire suite of Rooms.

2.1.2 Activity-Based Computing system

Bardram et al. (2006) [5] created a conceptual framework around the concept of "Activity", but didn't relate it directly to activity theory. The essential principles they defined are shown in Table 2.2. Like Card et al. [10] they identified the need to be able to switch between tasks (Activity-Centered and Activity Suspend and Resume principles) and task-specific presentations (Activity Adaptation and Context-awareness principles). With the rise of ubiquitous computing two extra principles were added, relating to the ability to roam and share activities (Activity Roaming and Activity Sharing principle). This allows for collaborative work on shared activities. Roaming isn't limited to roaming between desktop computers, but work can be continued (and shared) on any device, e.g. a tablet computer.

Activity-Centered A 'Computational Activity' collects in a coherent set a range of services and data needed to support a user carrying out some kind of (work) activity. This principle addresses the challenge of application-centered computing.
Activity Suspend and Resume A user participates in several activities and he or she can alternate between these by suspending one activity and resuming another. Resuming an activity will bring forth all the services and data which are part of the user's activity. This principle addresses the lack of support for interruptions.
Activity Roaming An activity is stored in an infrastructure (e.g. a server) and can be distributed across a network. Hence, an activity can be suspended on one workstation and resumed on another in a different place. This principle addresses the challenge of mobility.
Activity Adaptation An activity adapts to the resources available on the device (i.e. computer) on which it is resumed. Such resources are e.g. the network bandwidth, CPU, or display on a given devices. This principle addresses the challenge of isolated and homogeneous devices.
Activity Sharing An activity is shared among collaborating users. It has a list of participants who can access and manipulate the activity. Consequently, all participants of an activity can resume it and continue the work of another user. Furthermore, if two or more users resume the same activity at the same time on different devices, they will be notified and if their devices support it, they will engage in an on-line, real-time desktop conference. This principle addresses the challenge of collaboration.
Context-awareness An activity is context-aware, i.e. it is able to adapt and adjust itself according to its usage context. Context-awareness can be used for adapting the user interface according to the user's current work situation – or it can be used in a more technical sense, where the execution of an activity, and its discovery of services, is adjusted to the resources available in its proximity. This principle addresses the challenge of context insensitivity.

Table 2.2: Essential principles by Bardram et al.

Like Rooms, they also adopt the approach of grouping application windows together in activities, which are essentially virtual desktops. All activity operations can be found on an Activity Bar which replaces the taskbar from Windows XP, for which their system was developed. An "Activities" button is used to list the current user's activities. A few action buttons allows managing them. Frequently used activities are shown on the bar itself, allow-

ing quick access to them without having to go through the Activity button. Lastly the bar contains a few status icons mainly used for collaborative features. To support different display sizes and thus supporting the principle of activity roaming the user interface allows zooming in and out of an activity. Windows can be arranged on a larger 2D surface area than visible when zoomed in. In order to navigate within an activity, a radar view allows dragging around a red rectangle across an overview, moving the viewport around. They created an architecture which allows for the state of activities and its containing applications to be saved and restored. In order to make an application fully stateful so that it's entire content can be restored a special-purpose ABC application wrapper needs to be created for it.

2.1.3 Giornata

Voida et al. (2008) [39] created an activity-based desktop interface prototype, Giornata, informed by activity theory. The virtual desktop metaphor of Rooms was used as a starting point. They defined seven requirements, listed in Table 2.3, which overall show a heavy emphasis on usability of the system; providing the user the functionality of working activity-based without being too intrusive. To prevent interaction overhead as much as possible, using dedicated project spaces is formulated as a requirement, rather than having the unified activity model embedded in a single application.

Requirement 1 To integrate into existing work practice, activity-based systems should provide a unified activity model across all applications, rather than being embedded in a single application.
Requirement 2 Activity-based systems should provide lightweight mechanisms to create, change, and alter activities, since heavyweight interaction techniques are likely to deter adoption and use.
Requirement 3 Activity-based systems should provide tools for informally and formally organizing disparate information within activities. Informal information organization tools should emphasize quick storage and retrieval, without forcing people to explicitly name or find a permanent place for artifacts; formal mechanisms should correspond to long-term storage and retrieval practices.
Requirement 4 Real-world activities "overlap" in the way they use artifacts; a given artifact may be used in multiple contexts. Activity-based systems' representations of activity should support this overlap, rather than prescribing that activities be orthogonal or that their artifacts exist in only one context.
Requirement 5 Activity-based systems should allow post hoc definition of activities, enabling individuals to map their evolving understanding of the activities into the system; individuals should be able to create initially unnamed activities and then refine them after the fact. Artifacts used in unnamed activities may need to acquire these refined declarations of use as these activities evolve.
Requirement 6 Activities in activity-based systems should be usable as structuring mechanisms for collaboration (i.e., an activity-based perspective should be integrated into common collaborative tools).
Requirement 7 Because information sharing is a common case in knowledge work, lightweight sharing capabilities should be integrated directly as a first-class interaction technique.

Table 2.3: Requirements as defined by Voida et al.

Chapter 3

Using Activity Theory for Interaction Design

The Association for Computing Machinery (ACM) defines HCI as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them". HCI has its roots in a wide set of different fields; most notably human factors (HF), cognitive psychology and the systems part of computer science. It distinguishes itself from HF as the focus lies on humans working with computers specifically. Aspects of cognitive psychology are used to analyze perception, learning, performance and mental representations by humans of the designed systems.

As we are designing a system to support any type of knowledge worker in managing their work in a set of activities when using a computer, HCI is deemed to be the suitable field in which to look for a theoretical foundation. However, there doesn't seem to exist a "well-established body of harmonious scientific knowledge covering the basic foundation of the discipline". [27] On the contrary, even to this day there seems to be a mismatch between HCI research and interaction design practices. [19] Although HCI research aims at influencing interaction design and its practitioners it falls short in doing so.

Kaptelinin highlights two mainstream approaches to interaction design over the course of history. [25] A cognitivist approach which is based on a well-developed and highly structured conceptual framework that allows for generalizable models but suffers from its narrow scope, and ethnomethodology which provides rich depictions of practice, but fails at generalizing accounts or relating them to actual designer concerns. Both approaches have often been contested, and researchers are still on the lookout for a new theoretical framework.

3.1 Cognitive approach

Cognitive science has had great success in the past in shaping and evaluating graphical user interfaces, much of which originated at Xerox PARC. It helps in developing principles, guidelines, models and frameworks. This allows to conduct experiments which study factors relating to UIs, with a particular focus on usability.

Despite its success, problems have been recognized with the cognitive approach not too long after it was first introduced. A seminal paper in this ongoing discussion is Bannon's "From Human Factors to Human Actors". [3] HF often regards the human as another system component with certain characteristics which are analyzed within the overall human-machine system. This information-processing paradigm de-emphasizes or even neglects important aspects of work design like individual motivation, membership in a community and the the setting in which work occurs. Another major concern is human action has been observed to be "situated", or ad-hoc. Suchman argued that resources of the immediate situation shape human action. People are improvisatory. [37] She demonstrated this in a famous experiment where copy-machine users were shown *not* to be following algorithmic plans while struggling to make copies. This highlights a fundamental contradiction with cognitive science as it is usually applied to HCI research where humans and computers are seen as equal.

3.2 Ethnomethodology

Suchman's work in situated action had developed from an anti-theoretical branch of sociology, called ethnomethodology. Ethnomethodology's goal is to document the methods and practices through which society's members make sense of their world. [17] In practice, specific instances of organized action are analyzed and described in detail. Sociological theorizing is completely rejected. This results in it being perceived as rebellious by mainstream sociology, which adopted the view that purely descriptive work "must be judged a failure." [8]

Abandoning theory completely doesn't allow us to form a community with shared concepts which can be used to summarize, shape, compare or abstract what we learn. Theory provides us with an overview which can help us in making strategic decisions on how to proceed. It allows us to more easily have conversations, and juxtapose different points of view. Ethnomethodology does not provide interaction design with the means to theorize about what situated action claims to be missing – the social and contextual aspect of

human activity. [25]

3.3 Activity Theory

Activity Theory (AT) is gaining use in theoretical discussions on interaction design. [25, 5, 27, 38] It provides a middle ground solution between the two extremes; cognitive science and ethnomethodology. Although its name might imply otherwise, it's not a theory, but rather a descriptive framework. Within this psychological framework, activity is placed central when trying to understand the way how people act. Activity is understood as a purposeful interaction of the subject with the world, in which technology is viewed as a tool by which interaction is mediated. [25]

Since this thesis originally set out to find methods by which to support users in managing their activities, activity theory was quickly recognized as a possible framework to use during the design. It makes sense to use a framework which provides an in-depth description of activities, and makes them the focal point of analysis. Opposed to the human computer model, the scope of analysis needs to be extended to a meaningful context of a subject's interaction with the world, including the social context. Traditional task analysis as applied in human factors sets out to analyze one particular task in depth, decomposing it into a hierarchical representation of what steps to take to perform it. We feel our goal to support many kinds of knowledge workers in managing their activities in a wide possible range of scenarios and environments would suffer from the outlined problems when analyzing it with the traditional cognitive approach. We can learn from Suchman's concept of situated action by accepting that users respond opportunistically and flexibly to the immediate available resources. However, we do want a framework in which to frame our research, which is where activity theory comes in.

3.3.1 Activity system

The basic unit of analysis in activity theory (AT) is "activity", which Leontiev originally defined as "the nonadditive, molar unit of life for the material, corporeal subject". [30] In its narrowest sense, activity is a unit of subject-object interaction defined by the subject's motive. A motive is an object that meets a certain need of the subject. Objects are not necessarily physical things, they can be intangible. When a need is coupled with an object, an activity arises. [25] This is a principle called *object-orientedness*.

Activity has an hierarchical structure, and can be analyzed at different levels: *activities*, *actions* and *operations*. [30]

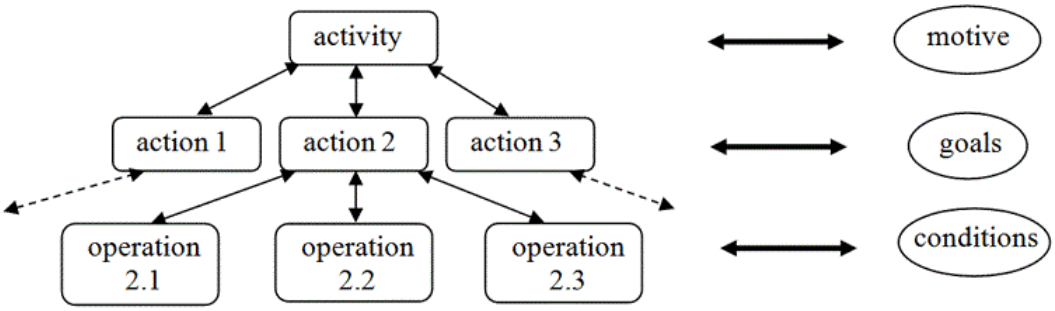

Figure 3.1: The hierarchical structure of activity. (source: interaction-design.org)

The top level is activity itself, oriented towards its motive. At a lower level lie conscious goal-directed actions that must be undertaken to fulfill the object. Goals can be decomposed into sub-goals, and so forth, meaning actions can have a hierarchical structure of their own. At the lowest level lie automatic processes which happen subconsciously, called *operations*. This is how actions are eventually carried out.

AT has come a long way since Leontiev first created its foundations in 1978. [31] Engeström expanded on the subject-object activity system by emphasizing the community in which activity occurs. From this resulted a model which also incorporates mediational means for the three-way interaction between subject, object and community, and an outcome of the activity system as a whole which can be used by other activity systems.

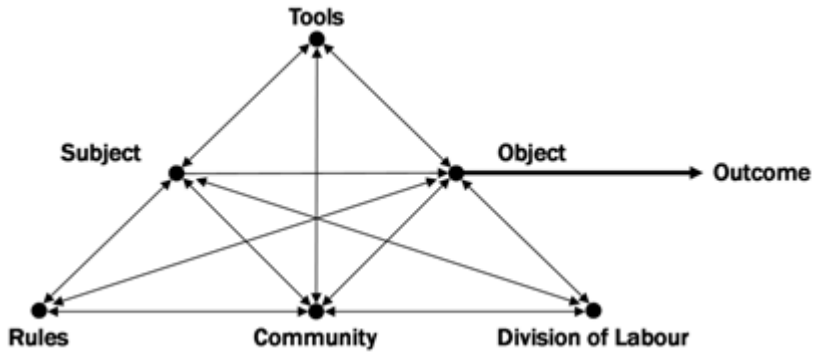

Figure 3.2: Activity System Model by Engeström. (source: http://www.iva.dk)

He described this framework as an

> object-oriented, collective, and culturally mediated human activity, or activity system. Minimum elements of this system include the object, subject, mediating artifacts (signs and tools), rules, community, and division of labor. [13]

Activity system components [16]

- **Subject** The individual or sub-group whose agency is chosen as the point of view in the analysis.
- **Object and tools** The "raw material" or "problem space" at which the activity is directed and which is molded and transformed into outcomes with the help of physical and symbolic, external and internal mediating instruments, including both tools and signs.
- **Community** Comprises multiple individuals and/or sub-groups who share the same general object and who construct themselves as distinct from other communities.
- **Division of labor** Both the horizontal division of tasks between the members of the community and the vertical division of power and status.
- **Rules** The explicit and implicit regulations, norms and conventions that constrain actions and interactions within the activity system.

Example in terms of knowledge workers

Consider the work activity of a knowledge worker (subject) on a computer in a company. Separate distinct tasks can be carried out, which each are aimed at a different outcome. (object) Several applications are available to aid the knowledge workers in their tasks. (tools) Depending on the work environment work might be shared across different subjects. (community) Work is divided both horizontally, as different specialists focus on different parts of the overall object, as well as vertically, reflecting the hierarchy of the company. (division of labor) To this end a certain management methodology might be followed. (rules)

Work can be performed on several projects (activities). To complete the overall project, several intermediate steps need to completed. (actions) While carrying out work the knowledge worker generally learns to automate some processes, like typing. (operations)

3.3.2 Social and developmental aspect

Three main principles of AT have already been mentioned during the discussion of the activity system model; *object-orientedness*, the *hierarchical structure of activity* and in short *tool mediation*. The activity system model is not fixed or isolated. AT regards social and developmental aspects as essential when analyzing activity. Development continuously reforms and develops practice.

Internalization and externalization Internalization and externalization are processes that relate the human mind to its social and cultural environment. [25] It involves two dimensions.

- Physical: internal (mental processes) $\longleftrightarrow$ external behavior
- Social: intrapsychological (individual) $\longleftrightarrow$ interpsychological (collective)

Internalization is the transformation of external activities into internal activities. It causes a redistribution between external and internal components of activity, and in some cases, some external components can be omitted entirely. [25] E.g. when learning to touch type, the external act of looking at the keys after a while *transforms* into being able to type without looking at the keyboard.

Externalization transforms internal activities into external ones. This is often necessary when internalized actions need to be "repaired", scaled or performed externally so they can be coordinated.

Similar transformations occur along the social dimension, producing similar outcomes. AT emphasizes that it is the constant transformation between the external and internal that is the basis of activity. [25]

3.3.3 Contradictions

Activities aren't isolated but should be seen within a network of other activity systems which have an influence on each other. Change to activities arises from the incompatibilities, conflicts or opportunities, known as *contradictions* (also called breakdowns), that can exist both within and between activities. [38] This can also result in the end of an activity, or give rise to new ones. Four different kinds of contradictions can be identified, based on where they occur.

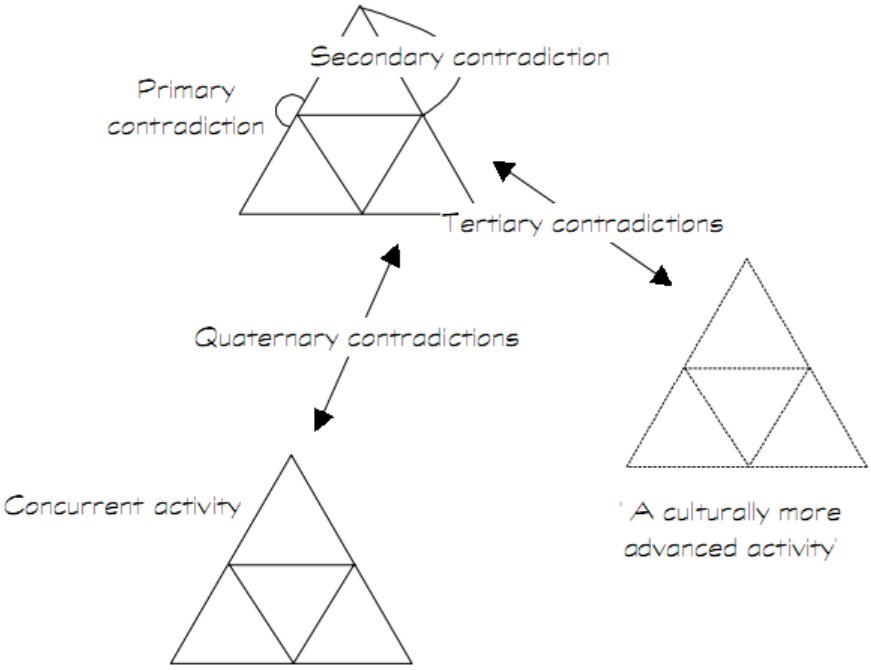

Figure 3.3: Four levels of contradiction. [38]

Primary contradictions are found within a single element (node) of the activity system; *secondary contradictions* between two different nodes; *tertiary contradictions* between an activity and a new form of the activity with a "culturally more advanced" object; and *quaternary contradictions* between the central activity and its neighboring activities.

Contradictions are not to be viewed as negative, but as opportunities for development. By seeking out contradictions within an activity system you can look for ways in which the current system can be improved. This is a contradictions-driven approach to analyzing the activity system.

Chapter 4

System Design

Prior to designing the prototype, the entire set of conceptual requirements which an ABC system should support are analyzed. This allows us to make an informed decision on which requirements we should focus, as it is impossible to discuss them all within the scope of this thesis. It also allows us to see the bigger picture, and design a user interface in which the other requirements could be incorporated later on. Those requirements which contribute most to existing research are selected and used as a guideline when designing the prototype.

4.1 Conceptual requirements

Previous research identified a set of conceptual requirements for developing an interface which supports the user in organizing work in a distinct set of activities. (See Section 2) Most of those requirements were brought forward as to answer observed shortcomings of the current prevalent desktop metaphor. In order to contribute to this set of requirements we decided to adopt a different approach. First we reviewed all previously proposed requirements, finding commonalities and structuring them into one model. Afterwards by analyzing what constitutes an "activity", along with identifying gaps within the model, new requirements suitable for an ABC system were added. Intentionally broad descriptions are used, as the requirements mainly act as a tool to subdivide the problem space into different segments which can be studied more thoroughly later on.

Describing concrete scenarios of how users would interact with a UI incorporating these requirements helps in evaluating them, and is a first step towards determining what such an interface would look like.

4.1.1 Requirements analysis using activity theory

Using Engeström's Activity System Model (see Figure 3.2) requirements can be analyzed within the context of activity theory. Taking into account the entire activity context and the relationships between the different components helps in identifying requirements which can support subjects in their activities. A summary of all identified requirements can be found in Table 4.1.

Each node (dot) in Engeström's triangle is a suitable perspective to analyze an activity from. Doing so can lead to insights in understanding where current user interfaces do not support the user in managing multiple activities. A contradiction-driven approach to requirements analysis is followed as specified by Turner. [38] We are mainly looking for secondary (inter-node) and quaternary (inter-activity) contradictions. The requirements could be listed by type of contradiction they attempt to resolve, but we found listing them under the most suitable node gives a better overview and allows grouping several contradictions together more easily.

Subject perspective

- **Managing activities** The main requirement originates from the concept of activity itself. The system design should reflect the object-orientedness of knowledge workers. An overview of past, present and future planned activities needs to be available, offloading some of the mental load required to keep track of them. Switching between them should be fast, and only the relevant activity context needs to be visible at any given time. Activities are dynamic, and can change over time, arise from existing activities or merge with each other. Visualizing this activity hierarchy over time can help in identifying activities.

- **Information access** Knowledge workers need to be able to access information from a wide range of sources. In order to guarantee a productive environment, all their existing ICT tools need to be supported. Although all data should still be accessible, reducing the immediately available data to just the sources required to achieve the current objective reduces information overload. External resources can be accessed more efficiently by searching within specific communities, relevant to the activity. Finally, access to certain resources could be restricted based on rules, or scoped based on division of labour.

- **Mental model** While working on an activity users form a mental model of the activity in their working memory. Over time this mental model is lost. The system needs to support the user in storing and

restoring it. This can additionally help in transferring working context between users. (externalization)

- **Interruption handling** Interruptions can originate from the entire activity context. Tools report their status, objects change or you receive messages from collaborators. Other interruptions, not related to the current activity might be of importance too. Providing the user with a central location for interruption handling can help in reducing the time needed to track different possible sources independently. They can also be visualized in time. Interruptions originating from the current activity can be given higher priority. Interruptions can give rise to new activities, or the user might want to switch to another activity.

- **Ubiquity** An activity shouldn't be accessible just from one location. The activity and its entire context should be able to roam from one device to another.

Community perspective

- **Activity sharing** Multiple participants can carry out (simultaneous) work on a shared activity context. Activity context-aware collaboration tools can simplify sharing resources, finding relevant participants, and prioritize incoming messages. A subset of an activity can be shared in order to allow division of labor, and restrict access to the wider set of resources. Each participant in the activity can control their own view of it.

- **Conventions** Common conventions or procedures can be reinforced by providing subjects with pre-initialized (also see Templating) activities from which they can start work. Entire collaboration processes can be templated to speed up the initial setup of a work environment.

Tools perspective

- **Tools plasticity** Tools should act as mediators to aid in work on a given activity. The tools need to be adaptable to the subject's preferences in order to facilitate internalization. The needs can vary based on the activity at hand, or the resources of the current device the user is using. Preferences should be persisted per activity, so the user has the possibility to set up an optimal working environment to achieve his object. Tools should be extendable by the community so the tools can grow and adapt based on experiences during the development of concrete activities.

- **Interoperability** As ubiquitous computing becomes more prevalent, work on one activity context should be able to be divided across several devices.

Object perspective

- **Templating (externalization)** Common activities could be templated so they no longer need to be initialized each time. This template can contain the relevant subjects, tools, etc. required to commence work on the activity.

- **Activity discovery** Objects that don't have any subjects assigned to them are possible sources for activities. One subject can discover an object and share it with the community. According to rules or division of labor activities can be formed by linking the objects to relevant subjects. Instead of having project management be an external tool it can be integrated within the system, directly linking it to the emergence of activities.

- **Resolve conflicts** Different activity contexts can have overlapping resources. In order to resolve these conflicts the system needs to warn the user when they occur, and offer a solution to handle them.

Subject	
Managing activities	Time
	Low overhead
Information access	Activity context
	Support existing tools
	Community integration
	Access restrictions
Mental model	Storing and restoring
Interruption handling	Centralized handling
	Prioritizing
	Activity emergence/switch
Ubiquity	Activity roaming
	Cross-device activity
Community	
Activity sharing	Simultaneous
	Shared context
	Division of labor
Conventions	Templated procedures
Tools	
Tools plasticity	Adaptable
	Persistent preferences
	Extendable
Interoperability	Cross-device activity
Object	
Templating	Preset work environment
Activity discovery	Project management
Resolve conflicts	Warn and handle

Table 4.1: Summary of conceptual requirements per perspective

4.1.2 Example scenarios

Considering real-world examples of how such a system can support different knowledge workers in day to day work is an important step towards determining what such a system should look like. The context of activities can be described in two important dimensions.

- *Cooperation*: an activity can either be mainly individual work, or cooperative work in teams.
- *Mobility*: work can be done in a fixed environment, or from several different devices. For the purpose of this dicussion, using the same

device from different locations (e.g. a laptop) isn't considered mobile.

Within these two dimensions, different knowledge workers can be situated in terms of the main activities they perform. Just because an item is situated near one end of a dimension doesn't mean it lies on this extreme. Almost any work involves activities which can be situated anywhere on the plane. Figure 4.1 places a manager, novelist, programmer and Personal Information Management (PIM) within these dimensions. These examples will be used to represent the corresponding area by describing a realistic scenario for them.

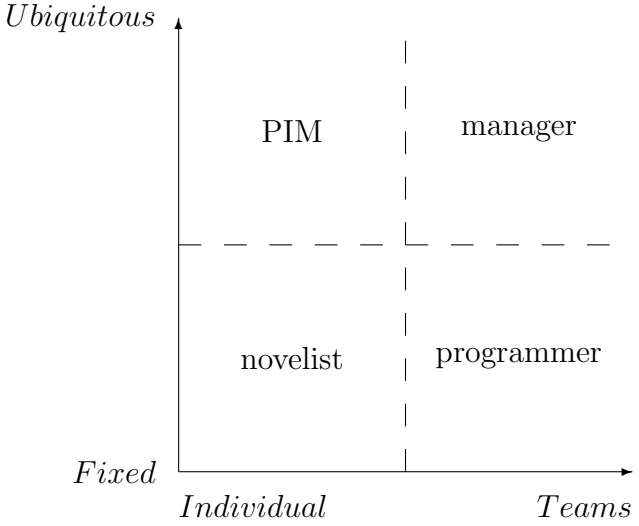

Figure 4.1: Activity dimensions

Personal Information Management

Even in a nonprofessional setting, an activity-centric way of managing information can be beneficial. There is a lot of information to manage: personal contacts, appointments and homemade as well as external multimedia. Although PIM is located near the individual part of the cooperation dimension, that doesn't mean people don't share information. However, privacy is a big issue and people want to control who has access to what. This scenario is described separately, but it is an integral part of computer activity, and thus the other described scenarios as well.

Scenario On a Friday evening while browsing the Internet, a user notices he received an instant message from a friend in the central notification system. A quick preview shows that he wants to meet up for a concert the next day. The user is interested, so decides to send the conversation to a new activity. All the applications and documents he was currently working on

disappear, and a new virtual desktop shows up containing just the instant message window. After talking with his friend he finds out the concert will start at 8PM, so he opens up the activity overview screen which shows a time line on which the newly created activity is visible, along with the previous browsing activity. He drags the newly created activity over to the next day, 8PM, and shares it with his friend. After re-opening the activity by clicking on it, a part of the desktop shows an area in which documents can be shared. They decide that while one person buys the tickets on-line, the other tries to find out how to get to the concert. While one user browses for route information, a notification appears after the other user dragged the PDF file containing the concert tickets into the shared documents area. This notification has higher priority than notifications originating from other activities, as it originates from the currently active activity. After everything is arranged, both users close the activity, returning to the time line to select any other activity they were working on.

The next day at the concert, the user sees a similar overview of his activities on his cellphone. He selects the concert activity and starts taking pictures, which are automatically assigned to the concert activity. When arriving home after the concert, all the pictures taken are also available on his desktop computer under the relevant activity.

Novelist

Some artists work mainly individually on the piece they want to create. A novelist can spend several months on his own working on his laptop to finish a book. Although an ABC system doesn't reach its full potential in such a setting, it can still help in organizing and planning work.

Scenario A novelist just finished writing a chapter, and she decides to go to the activity overview screen, which shows an overview of the coming week. Gradually zooming out, she starts to see the bigger picture of her progress. In three months she is supposed to hand in the first draft of the book to her publisher. She still has three chapters to write. For each one of them, she has assigned a deadline, visible on the time line. Still needing inspiration for the setting of her last chapter, she opens the relevant activity and starts browsing for images on the Internet. Since her tablet computer is set up to have the same activity open as her desktop computer, it now also shows just those items relevant to the currently selected chapter. The screen saver of the tablet PC is set to display images from within the activity in order to inspire her. This behavior is the same for every chapter activity, as they were created from the same preset.

Coming across some nice typography on the web, she is reminded she still has to choose a font for the chapter headings. When switching over to an activity she created especially to do all lay out work, she is presented with the same text editor she usually does her writing in, but specifically set up for the task at hand. Her tablet computer now shows a preview of the document instead of the images.

Having chosen a nice font, she decides to sit down on the couch with the tablet computer. She switches back to the activity of the chapter she just finished, starts reading and making amendments as she goes.

Programmer

Programmers usually work in teams towards a shared goal. Working from home can occur, but in general work is done from a fixed location. Work needs to be coordinated based on requirements from the client, designers and fellow programmers.

Scenario While working on a new feature, a developer notices an incoming bug report assigned to his name through the central notification system. The project lead assigned it to him since he is the one who did the implementation. As the bug reporting tool is activity-aware, the programmer can easily click a link to take him directly to the original activity which was created when the feature was first requested. Under the shared section of the desktop he can find the feature documentation which was delivered to him over a month ago. The relevant source files which he last accessed when implementing the feature are visible in his programming environment. As he is unable to reproduce the bug he requests the log file associated with the crash, opens up his previous activity and continues his previous work.

Later that day he receives a notification to accept a new shared activity. Its details indicate it's a newly created sub-activity of the feature activity, containing the log file. Opening it brings him to an empty desktop with just the shared log file visible. The bug report is still located in the feature activity, so he decides to go to the activity overview screen. This now shows a relation between the original activity and the newly created one. He is able to identify the bug report in the preview of the original activity, and drag it over to the new one. Looking at the log he notices an exception message which seems to originate from an in-house library. By doing a community search he can search through the company's documentation, finding the information he needs to resolve the bug.

Manager

Out of the previously described scenarios, a manager most likely has to coordinate with the largest amount of people, and is often on the road. While an ABC system can be used to its fullest extent in such a setting, resistance to adopting the system might be highest. To streamline work, managers already use a wide array of planning and project tools. All these would have to be tightly integrated into the system, or be replaced by it.

Scenario A project manager is preparing a presentation to present to the client later that day. Part of it is a progress report of the project. She decides to take a look at how her team is doing. After switching over to the activity time line, she selects the implementation activity, and the interface zooms in to show a new time line with all the separate features laid out over time as activities. An indication shows one of the features which was supposed to be finished is still active. In the details of the activity she can see the assigned programmer, allowing her to quickly contact him for a status report. An unexpected problem will delay the feature significantly, so an emergency meeting is arranged. The entire team receives a high priority meeting request notification, which shows up as a shared activity on their activity time line.

During the meeting the manager dismisses a call she received on her cellphone. Later, she sees a notification of this on the time line at the exact time of the call. Her boss was asking for the previous meeting report, which she can simply find by accessing the meeting activity of last month on her time line. After the meeting the manager merges the meeting activity with her presentation activity, in order to fluently incorporate her notes into the presentation. An incoming notification tells her she has to interview a job candidate in 30 minutes. Clicking the notification immediately takes her to the activity which triggered it, allowing her to review the candidate's CV which is visible there, before heading to the interview.

During the interview she gets a notification on her cellphone of the presentation she has to give in 30 minutes. She finishes up the interview and heads directly to the reserved meeting room. She can access her presentation from her cellphone through the meeting activity, and uploads it to the old presentation PC which doesn't support the activity-based system yet.

4.2 Interface design

Based on the conceptual requirements, more concrete interface requirements can be formulated. The scope of a complete ABC system is too wide for this thesis, so only a subset of the requirements are taken into account while designing the user interface. The emphasis is on those requirements that continue on the open issues of previous research; managing activities in time and space and centralized interruption handling. Table 4.2 lists the selected requirements.

Subject	
Managing activities	Time
	Low overhead
Information access	Activity context
	Support existing tools
Interruption handling	Centralized handling
	Prioritizing
	Activity emergence/switch
Tools	
Tools plasticity	Persistent preferences
Object	
Templating	Preset work environment
Resolve conflicts	Warn and handle

Table 4.2: Subset of conceptual requirements

Two main components can be identified when designing an ABC system. In order to manage activities you need an overview of them. This thesis proposes an overview of activities in time and space. A similar way to organize information has been explored before, e.g. Lifestreams [14] and TimeSpace [26]. Providing an overview of activity context over time can be a powerful cue for recall by invoking episodic memory. [28] Once working on a particular activity, only the relevant activity context should be visible, which can help in reducing information overload. (dedicated project spaces) By setting up a virtual desktop per activity this can be achieved, while still supporting all the existing tools the user is used to. For each conceptual requirement, supporting interface requirements will be specified.

4.2.1 Activity overview

The activity overview screen is a full screen visualization from where all activities can be accessed and managed. The main component on this screen is

a time line on which all activities are laid out. The position on the time line indicates when activities were active or are planned, while spatial visualizations can indicate relationships between them. Interruptions are visualized on the time line as well. The overview screen also shows a clock, calendar and search box.

Managing activities – Time

Create new activities New activities can be created from anywhere in the system instantly. After doing so, the new (empty) activity context becomes visible. By default, the activity is placed at the current time on the time line. Activities can also be created from the time line, allowing to immediately specify where in time the activity should be placed.

Open and close activities The entire activity context of an activity can be closed, and later restored to its exact previous state. Closing an activity frees up its computational resources. Reopening an activity can take some time since these resources need to be restored.

Switch between activities Switching between different activities should be fast and fluent, allowing the user to easily switch between different work contexts. Opening an activity is possible by clicking on it on the time line. Computational resources aren't freed up in order to allow for fast resumption. When an activity is visible, only the relevant applications and documents for that activity should be visible. Each activity is represented as a virtual desktop.

Remove activities Ordinarily activities aren't meant to be removed, as they serve as a way to access information over time. However, removing activities is a requirement in order to allow users to remove accidentally created activities, guarantee privacy, or just to keep their time line clean.

Plan activities Activities can be created as a placeholder for work which needs to be done in the future. A specified amount of time can be reserved, but is not required. Planned activities are visible on the time line at their desired location. The reserved time can be changed by expanding or contracting the activity along the time line. While active or passed activities by default can't be reorganized in time, future activities can be freely dragged and dropped to a new location. Recurring activities can be added, with a fixed interval over a specified amount of time.

Merge activities Different activities can be merged together into one activity when it makes more sense for the activity contexts to be shared. The original activities on the time line are closed, and their context is transferred to a newly created activity.

Manage activities on a time line An activity should have a visual representation on a time line, indicating its status over time. It is visible when an activity was open, or for future activities when they are planned to be open. When reopening an activity it shows up at the current time on the time line, but a marker allows to jump back to the previous time the activity was open. A similar marker allows to jump forward in time to where the activity is reopened. Any activity can be relocated in time, although this is locked by default for all activities but those planned to start in the future.

Focus of attention An optional line on the time line indicates the focus of attention of the user over time, going from activity to activity. When no activity, or a planned (future) activity is active, the line is drawn along the time line, otherwise it "enters" the activity representation where the user switches to it, and leaves it when the user switches back. Additionally, the latest active activity is always highlighted.

Clock and calendar A clock and a small calendar are visible on the same screen as the time line. They mainly act as a reference for the user when planning activities. The calendar can also be used to jump to specific days directly. When holding the mouse at any given point on the time line, the exact time at that position is visible, as well as the relative time to the current time.

Manage activities in space Activities are placed in a two-dimensional space. One dimension represents time, while the other can be used to visually represent activities and their interdependencies. The user can resize and position activities in this dimension in order to assign importance/categorization to them. Lines between activities can be drawn to indicate relations between them. These lines are created automatically when activities are merged, or when an activity is split from another.

Creating sub-activities Activities can be grouped together in a parent activity. A parent activity gets its own time line with only its sub-activities visible on it. Parent activities are visually distinguishable from ordinary activities. Opening them doesn't open a virtual desktop, but opens their time

line. Activities can be drag and dropped on the parent activity in order to add them. An empty parent activity can be created from the time line through a context menu. Alternatively one can be created through the context menu of an existing activity, moving it within the newly created parent activity, replacing it on the time line. A parent activity always requires a name. This name is used as a hierarchical "breadcrumb" presentation at the top of the time line. Clicking on these breadcrumbs allows returning to higher hierarchies.

Tagging activities Tags (short textual meta-data) can be assigned to activities in order to classify them. Existing tags are proposed through auto-completion to encourage reusing similar tags when appropriate.

Search activities A search box allows searching through all activities by using key words. Activity names, tags, context documents and applications are considered during the search. Results are shown in a list as well as on the time line. Hovering over an item in the list makes the time line scroll to that activity so that the summary becomes visible. When no activity is selected, the time line scrolls back to its previous position.

Filter activities The user can decide to only show a subset of all activities by setting up a tag filter. Only activities which contain one of the specified tags are visible.

Activity representation An activity has two separate representations on the time line. One is a default concise representation including just a name and an icon. Another is a larger representation when the user selects it, additionally including a screen shot of the last state of the virtual desktop, associated tags and a list of open documents. The detailed view of a parent activity lists all its sub-activities, acting as direct links to them.

Edit activity description From the time line, the activity icon and name can be edited by clicking on them. When clicking on the activity name the user can directly start typing a new activity name. Clicking on the icon opens a pop-up from where users can choose between a default set of icons, or load their own.

Managing activities – Low overhead

Post-hoc activity description An activity doesn't need to be defined when creating it. An activity can be created instantly and will show up on the time line afterwards with default values. The name of the primary document of the activity and the icon of the primary application are used, or an empty representation when none can be inferred. Once one of the descriptive elements are edited, the system will no longer automatically adjust them.

Gradual zoom Instead of supporting different zoom-levels for the time line, a fluent gradual zoom allows zooming from a level of detail of minutes to years. In between, hours, days, weeks and months can be recognized. Activities are resized in real-time to reflect the time they occupy on the time line, with a minimum size so they are always visible.

3D interface By applying perspective to the time line so that it goes into the background, more information can be shown on the same amount of screen space. Detail is lost, but a larger timespan of the time line can be displayed, improving the overview of the activities. The third dimension can also be used to visualize opening sub activities by zooming in and out of them, helping the user in understanding the relation between parent and sub activities.

Keyboard shortcut keys All possible actions are also accessible through keyboard shortcut keys. To improve navigation on the time line some additional shortcut keys are available; it is possible to traverse activities in order (previous/next) and to jump backwards and forwards in steps dependent on the current zoom level.

Interruption handling – Centralized handling

Application interruptions Interruptions are mainly triggered by applications. Since applications are context aware, they know whether or not an interruption belongs to a certain activity context. An application decides what is considered as an interruption and when to trigger it.

Activity interruptions Interruptions can also be triggered by planned activities. A reminder can be set when to notify the user prior to the planned starting time of the activity. When more time is spent on an activity than planned, an interruption is triggered notifying the user.

Interruptions in time Interruptions are visualized on the time line at the time when they occurred, represented by an icon. They can either be handled or unhandled. When they originated from within a certain activity context they are attached to the relevant activity. Unhandled activities are always shown, regardless of whether they occurred within the currently visible timespan. When outside of the currently visible scope, they are shown on the sides of the time line, pointing in the direction on the time line where the interruption did occur.

Handling an interruption An interruption is handled in one of the following ways:

- Opening the activity it originated from. This is only possible when it is bounded to an activity context.
- Creating a new activity for it. In case the interruption occurred from within an activity context, the newly created activity is visually linked to the activity from which it originated on the time line.
- Assigning it to an existing activity.
- Immediately discarding it. The interruption is still visible on the time line, but is indicated as handled.
- Postponing it. The priority of the interruption is lowered, so that other interruptions appear first.

Interruption preview Interruptions can be previewed prior to handling them. A small visual area can be filled by the application. This area can also be interactive in order to support quickly handling an interruption without opening it in an activity.

Interruption handling – Prioritizing

Unhandled interruptions are visually represented in order of importance when they lie outside of the visible time span. Irrelevant interruptions can be postponed, while new interruptions appear first.

4.2.2 Virtual activity desktop

When working on an activity, only the relevant activity context is immediately visible. This activity context is represented by the well-known traditional Windows 7 desktop, allowing users to work in the environment they

are used to. Instead of just having one desktop available, a virtual desktop is available per activity. Switching between them is possible through the previously described activity overview screen (Section 4.2.1). A list of open activities allows to switch directly to them without having to open the time line. Incoming interruptions are visualized in a central non-intrusive location as long as they are unhandled.

Managing activities – Time

Switch between activities A much denser overview of just the open activities is available from every virtual desktop associated per activity. No excess space is occupied to represent the activities in space and time, since this space can be put to better use to show information relevant to the open activity context. Only switching to already open activities is possible, represented by a simple list of icons. These icons, along with optional activity names, are the same ones as used on the activity time line.

Open and close activities No existing activities can be opened without switching to the time line. However, activities can be closed through the shortlist visible on the virtual desktops.

Split activities When a user feels the need to separate part of an open activity context into a new activity, this is possible by dragging the desired files and windows into an allocated area on the desktop. When finished, clicking a button moves the newly created activity context to a new activity which immediately shows up.

Managing activities – Low overhead

Keyboard shortcut keys Although a user interface can benefit from an interface which can be manipulated by easy intuitive mouse operations, there is still a need to allow easy access to all common operations through the keyboard. Caps Lock is a key which is barely used and can be put to better use as demonstrated in the Humanized Enso project where it serves as a way to enter commands which can be accessed from anywhere. [23] Borrowing this idea, Caps Lock is used in order to switch to the time line when released, or in combination with other keys to execute common operations.

Auto hide All the additional visualizations of the virtual activity desktop can optionally be auto hidden. Only a tray icon remains visible at all times,

indicating the amount of unhandled notifications. This way the user is disturbed as little as possible while working on his activity and more space on the screen is kept free. Interruptions trigger a short animation in the tray bar, along with a sound notification.

Information access – Activity context

The activity context when working on a virtual desktop is comprised of:

- Open windows and their location, including whether or not a window is minimized.
- Files which are placed on the desktop. All files placed on the desktop are only visible from within the activity context in which they were placed there.
- A list of recently opened files, in order of when they were last accessed.

Information access – Support existing tools

An activity and its context is represented as an ordinary Windows desktop, with just a few added features. It is possible to work with any Windows application as one ordinarily would, but only those applications which are integrated with the ABC system are guaranteed to restore a previous activity context correctly. Without integration into the ABC system it is impossible to reload a complex activity state, or take into account changes made to the state while the activity was closed. Furthermore, only ABC system integrated applications can trigger interruption notifications.

Interruption handling

All unhandled interruptions are shown in a small prioritized icon list. Handling and previewing interruptions is done in the same manner as on the activity time line.

Templating – Preset work environment

From within a virtual activity desktop, you can choose to store its state as a preset. Using this preset, new activities can be created which contain the entire activity context of the preset. Presets are accessible from the time line as well as from the tray icon visible on every virtual desktop. When creating a preset the user is required to enter a name which is used when listing the presets.

Tools plasticity – Persistent preferences

Activity-context aware applications can store preferences per activity. This allows setting up an application for a particular task under one activity, but setting it up differently for another one. These preferences aren't lost when switching between activities, neither are they lost when closing one.

Resolve conflicts – Warn and handle

The system can detect when a resource which is already used in another activity context is modified, moved or removed. When this happens the user has to decide how to handle the conflict. A list of all conflicting activities is presented. Activities with non-activity-aware applications using conflicted resources are marked as unhandleable. No proper handling is guaranteed, and errors might occur when reopening the activity. The user can choose to either cancel the operation, or ignore the warning. Activity-aware applications allow more advanced handling:

- Close the resource in the conflicting activity so it is no longer used.
- Make a backup of the resource. Both activities will work on separate files from then on. The backup resource is placed on the desktop of the conflicting activity.
- When a resource is modified the activity can take over editing rights, allowing any further edits. Only one activity can have editing privileges for a resource at a time.
- When a resource is moved, the user can choose to use the new location in the conflicting activities as well.

Chapter 5

Implementation

We set out to implement a minimum set of features from those described in Section 4.2, needed to evaluate the system during an in-field study. Our goal is to bring across the concept of Activity-Based Computing, without explaining it in detail. An ABC system is designed to be more aligned with how people actually structure their work, thus a qualitative analysis of the system can determine whether it succeeds in doing so.

The resulting application is called "Laevo". It's name is derived from a Latin verb which is commonly translated as: to raise, elevate, lift up; to make light, lighten; to relieve, ease, comfort; to mitigate, alleviate. We decided on this name, as it entails most of the practical and emotional goals we hope to achieve. This section will focus on some of the design decisions in detail, as well as some of the engineering challenges of ABC. An in-depth description of all the features which were eventually implemented can be found in the user manual. (Appendix A)

5.1 Design decisions

The time line is the "heart" of the application. It acts as an overview, as well as a staging zone for new activities. It should convey the feeling of working on a higher abstraction, taking the user out of their usual desktop environment. Therefore we chose to make the overview full-screen, hiding any work the user was working on earlier. We no longer want to reason in terms of applications, windows and documents, but in terms of activities.

Contrasted to that, when working on an activity, we want it to feel like you are working "in" this activity, your entire attention devoted to it. Only the relevant activity context is available. We do not want to impose an overhead on the user when doing what they are already used to, their work

environment remains practically unchanged. (Information access – Support existing tools requirement) The main difference is they have not one, but several "dedicated project spaces" available to them. (Information access – Activity context requirement)

5.1.1 The time line

The main function of the time line is providing an overview of activities and allowing to manage them. (Managing activities – Time requirement) Switching between different "working spheres" occurs often [18] and thus needs to be supported with a minimum amount of overhead. (Managing activities – Low overhead requirement) The overview needs to be accessible from anywhere, anytime, which is possible by dedicating a physical keyboard key to it. Caps Lock is deemed useless by many, and Jef Raskin actually proposed removing it. [35] Google's Chromebook went ahead and did exactly that. [20] Instead, we followed a similar approach as Humanized Enso [23], and re-purposed it to open up the time line.

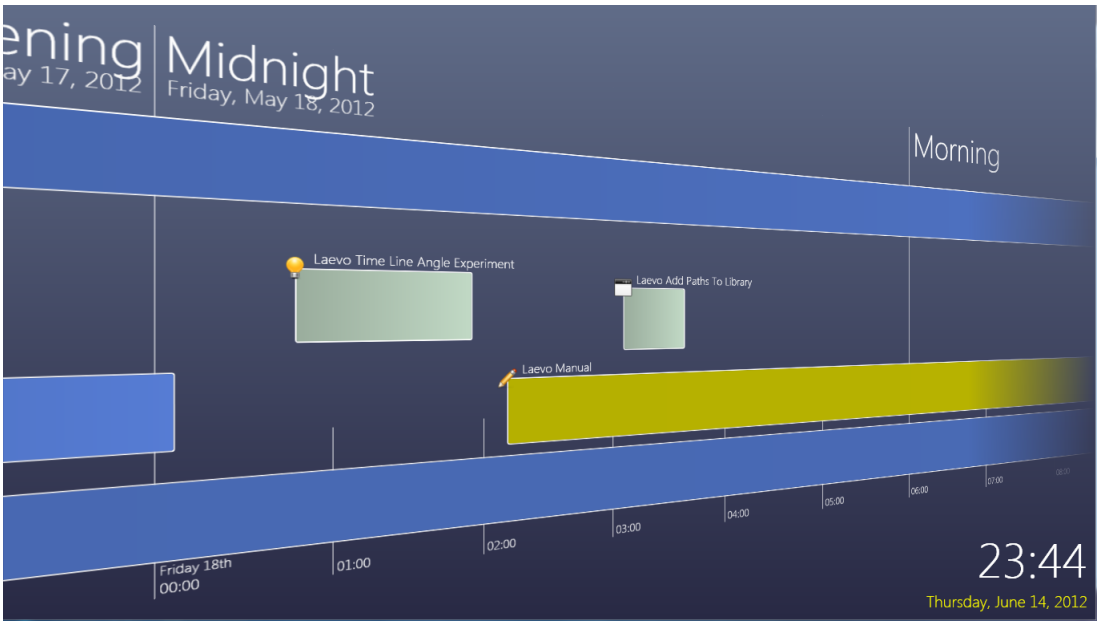

Figure 5.1: Activity time line

We defined a few key desirable properties around which most of the design is focused. The main concept of the time line was inspired by BeeDocs Timeline 3D for Mac OS X. [7]

The user needs to be able to view concurrent activities at any time scale. The granularity of activities can vary drastically; some lasting weeks, others mere minutes. Ordinary calendars often provide separate views of preset time spans: days, weeks, months. The activities on the time line are supposed to give an accurate representation of the time during which work was done on them. It is of high importance to visualize which activities happened concurrently. We feel it's easiest to provide this overview by representing activities in time along just one dimension, unlike e.g. a calendar overview of a month represented along two dimensions. This way the second dimension can be used to represent concurrency, and additionally even arbitrarily chosen semantics, like importance. Using one dimension also emphasizes the linearity of time. By using two simple manipulations – zooming and moving – the user is able to set up any visible time span which is most appropriate. E.g. there is no need to show activities in the past when planning an activity in the future.

Applying a perspective transformation has two advantages. It saves screen estate without hampering readability too much. Larsen et al. [29] obtained evidence that words whose first letters are more distorted than later letters are harder to read. As a resulting guideline they state "if only one side of a virtual environment is to be used for presenting text, the left wall is preferred". A second advantage of tilting the time line back is it strengthens the presentation of time. A common conceptual metaphor is expressing time as a physical path into space. The *Moving-Ego metaphor* refers to an observer moving frontwards towards a future. Common examples are "approaching the end of the year", or "leaving days behind". Conceptual metaphors shape not just our communication, but also influence human thought. Psychological research has demonstrated how real or imagined physical motion scenarios can prime construal descriptions of time by activating the relevant source domain. [33] The slanted plane primes the user in perceiving the far right side of the plane as "future", and the left side of the plane as "past".

This follows two of the display design principles, "Principle of pictorial realism" and "Principle of the moving part" as defined by Christopher Wickens et al. [40]

- **Principle of pictorial realism.** A display should look like the variable that it represents (e.g. high temperature on a thermometer shown as a higher vertical level). If there are multiple elements, they can be configured in a manner that looks like it would in the represented environment.
- **Principle of the moving part.** Moving elements should move in a pattern and direction compatible with the users mental model of how

it actually moves in the system. For example, the moving element on an altimeter should move upward with increasing altitude.

Minimize the discontinuity caused by viewing data at different times and places. When more information needs to be visualized than fits the screen, common techniques are paging or scrolling. This introduces a discontinuity between the information displayed at different times and places which can cause cognitive and mechanical burdens for users. [11] We opted for continuous zooming to address this issue. Ideally semantic zooming should be supported as well, but due to time restrictions we haven't gotten around to implementing this; when zooming out nearby activities could be grouped in lists in order to maintain a tidy overview.

To counter "desert fog", a condition wherein a view of an information world contains no information on which to base navigational decisions [24], new labels appear gradually when zooming in. The labels most appropriate at any given time are shown. See Figure 5.1 for an example where hourly labels are visible when the user is zoomed in to the "day level". Zooming in further, the hourly labels grow larger until they become the "dominant" labels shown at the top, and quarterly labels will be shown at the bottom instead. It doesn't make much sense to have an overview of activities at the minute level, so you can't zoom in further than quarters.

At a glance, it should be possible to see the scale, as well as the position in time of the information presented. Are we looking at an overview of hours, weeks, months or years? We set out to dedicate a big part of screen space to just this function, as without it you can't interpret the presented data correctly. Big "dominant" labels are shown along the top of the time line, showing the most important part of the date, as well as giving an indication of the current zoom level. Right underneath it the complete context is given by showing the full date.

This information can always be found at the top left corner of the screen. There is no need to track along the time line to find the first visible label. While moving the time line, the "earliest" top label stick to the screen sides until it is pushed of by the next one. (See Figure 5.1) This follows the "Minimizing information access cost" principle. [40]

- **Minimizing information access cost.** When the users attention is diverted from one location to another to access necessary information, there is an associated cost in time or effort. A display design should minimize this cost by allowing for frequently accessed sources to be

located at the nearest possible position. However, adequate legibility should not be sacrificed to reduce this cost.

The moment in time which is being viewed should be clearly contrasted with the current time. Yellow is used as a highlight color to indicate "now", as well as "active". A yellow line on the time line shows the current time; the currently active activity has a yellow border; in the bottom right corner the current date is shown in yellow; the attention lines which indicate when an activity was active are yellow.

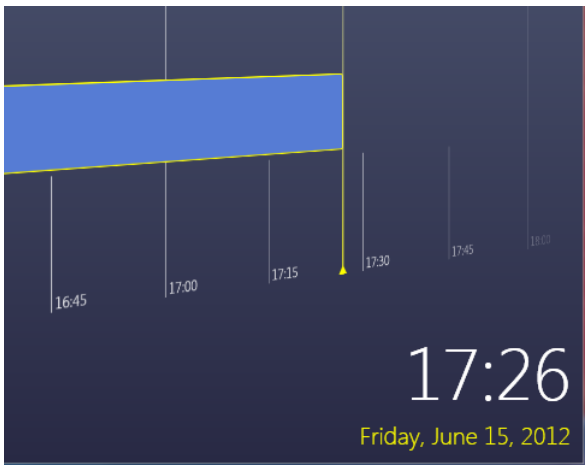

Figure 5.2: Current time visualization

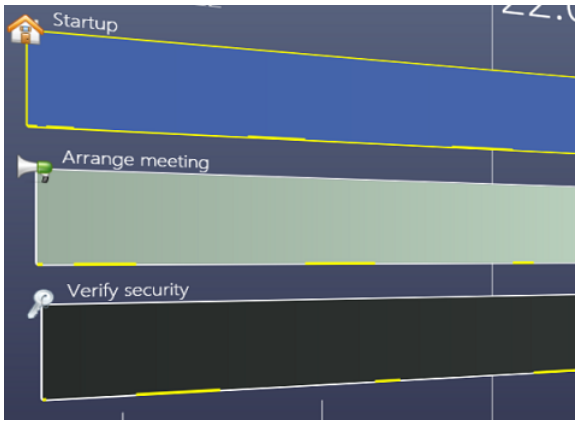

Figure 5.3: Attention lines

This follows the "Proximity compatibility principle". [40]

- **Proximity compatibility principle.** Divided attention between two information sources may be necessary for the completion of one task. These sources must be mentally integrated and are defined to have close mental proximity. Information access costs should be low, which can be achieved in many ways (e.g. proximity, linkage by common colors, patterns, shapes, etc.). However, close display proximity can be harmful by causing too much clutter.

Using a common color to represent "now" is only a first step towards following this principle. In our interface requirements (See Section 4.2.1) we specified a requirement which also attempts to improve close display proximity: "When holding the mouse at any given point on the time line, the exact time at that position is visible, as well as the relative time to the current time."

Activities should be easily recognizable. Activities are the main components visualized on the time line. They are central to an ABC system. Recognizing them is of the utmost importance. Therefore we decided to allow the user to set up several distinct visual cues for them. This follows the "Redundancy gain principle". [40]

- **Redundancy gain.** If a signal is presented more than once, it is more likely that it will be understood correctly. This can be done by presenting the signal in alternative physical forms (e.g. color and shape, voice and print, etc.), as redundancy does not imply repetition. A traffic light is a good example of redundancy, as color and position are redundant.

The user can assign a name, icon and color to each activity. Additionally they can drag the activities vertically, thus allowing to arrange them in space. This combined with the temporal aspect in which they are displayed gives plenty of possible visual cues to identify them.

Research has shown that most users prefer the control afforded by an adaptable approach to personalization rather than a system-controlled adaptive approach. [15] Unpredictable autonomous interface adaptations can easily reduce a systems usability. [34] As the system is currently implemented none of the possible visualizations are adaptive or automated, except the temporal aspect. Findlater et al. [15] suggested a mixed-initiative design in order to satisfy a wide range of users, where the system and the user both control some of the interaction. One possibility would be to have the system suggest a name, icon and color, without actually setting it.

5.1.2 The activity context

We find the least disruptive way in which to introduce the concept of activity into existing computing systems is by using dedicated project spaces which differ as little as possible from the work environment users are used to. (Information access – Support existing tools requirement) This way, as long as users are working on a certain activity, they don't need to worry about any additional concepts involved with ABC. This is reflected by our decision of practically hiding our system as a tray icon on the Windows 7 taskbar. Only after pressing Caps Lock the higher-level concept of activity becomes visible.

All the shortcut key commands start out by using the Caps Lock key. Once pressing Caps Lock we want to convey a notion of working on the context of activities instead of applications. Two of the most interesting shortcut keys are "Caps Lock - X" and" Caps Lock - V", with which you can cut and paste windows respectively. This was implemented as a quick way to allow moving windows between activities since there was no time to implement a graphical user interface for it. Although a user interface should still be provided for this action, we feel it succeeds in getting the job done, and is even more usable than other similar systems. E.g. pinning a window in order to move it between activities. We decided on "Caps Lock - X" and "Caps Lock - V" as this is consistent with the shortcut keys used to cut and paste text or documents, "Ctrl - X" and "Ctrl - V", following the "Principle of consistency". [40]

- **Principle of consistency.** Old habits from other displays will easily transfer to support processing of new displays if they are designed in a consistent manner. A users long-term memory will trigger actions that are expected to be appropriate. A design must accept this fact and utilize consistency among different displays.

Chapter 6

User Study

The main research question brought forward in this thesis is:

> *Can organizing activities in time and space simplify managing them and reduce information overload?*

By determining requirements and creating a concrete interface design from them, a prototype was created. In order to validate whether the proposed interface design actually results in simplified activity management and whether it helps in reducing information overload, a user study comprised of two parts is conducted. First, subjects are requested to familiarize themselves with the prototype by using it during a full day of work. The second shorter part of the user study consists of a comparative test where subjects perform a variety of assigned tasks using both the existing Windows 7 environment, and the prototype which augments it.

6.1 Experimental design

The experiment *seeks to analyse* the effect of organizing work in semantically meaningful activities, managed in time and space, *for the purpose of* exploring ways in which to improve computer interaction *with respect to* cognitive task load and productivity *from the point of view* of the user *in the context of* a wide variety of computer-aided tasks.

A long-term in-field study is preferred, but the experimental nature of the prototype poses some limitations. It is not feature-rich enough yet to fully support knowledge workers in their everyday work, nor is there enough time to do long-term tests. This prevents us from making measurements which require longer use of the interface, like how well the system supports the user in finding a month-old activity and continuing work on it. A middle

ground solution is to split the experiment into two parts, a one-day in-field test and a short comparative user study. The in-field test allows the user to get used to the prototype, and get their opinion about it afterwards through a questionnaire. While measuring long-term effects is not possible, it is possible to measure immediate effects of using the prototype during a short user study. This second part of the experiment is meant to measure productivity and cognitive task load during task switches, compared to a traditional interface.

6.1.1 Participants

Since a limited amount of time is available, only 19 subjects were tested. Unfortunately one test failed, and two candidates didn't fit the originally outlined requirements. All subjects are computer-literate and have extensive experience with Windows 7. They are proficient in English, so have no difficulties understanding the English interface or English texts which are part of the user study. Their profession requires them to use a computer throughout the day.

Some participants aren't able to participate in the first part of the study, but are willing to participate in the second part. Since finding test subjects for the more intensive second part is difficult, we opted to allow them to participate without using the prototype interface during a full day first. To compensate those users are given an extensive one hour personal introduction to the interface during which every function of the system is explained prior to starting the experiment.

11 people participated in both parts of the user study, one person only did part one, and 4 people only did part two. In total 12 people participated in part one, and 15 people in part two. Three results were ommitted.

In order to motivate the participants, a ranking of how well they performed compared to others is presented after the study. The overall winner receives two cinema tickets.

6.1.2 In-field test

Since the prototype does not support all features as described in section 4, it can't be used to its fullest extent in a real work environment yet. However, it does support the main concept of switching between activities and managing them on a time line, as discussed in section 5. A full description of the provided functionalities can be found in the user manual which every tester receives. (See appendix A) In order to get an impression of whether users find this functionality useful, they are requested to use the prototype during at least one full day in their normal work environment. In case they weren't

able to use the system properly throughout the day, they are requested to continue using it the next day. Afterwards they fill in the questionnaire. To conclude, a short interview allows the subject to discuss the prototype freely. This should have given the subjects sufficient time to learn to use the prototype, preparing them for the controlled experiment which takes place at another day.

Questionnaire

The questionnaire used to assess the usability of the prototype is based on the System Usability Scale (SUS). This is a simple, ten-item scale giving a global view of subjective assessments of usability. [9] It includes items with both positive and negative wording to minimize acquiescence and extreme response biases. Recent research found no evidence that the purported advantages of including negative and positive items in the questionnaire outweigh the disadvantages of mistakes and miscoding. [36] Therefore, an all positive version of the SUS is used. These modified questions are listed in Table 6.1. The response options, arranged from the left to right, are Strongly Disagree (1) to Strongly Agree (5).

1. I think that I would like to use the system frequently.
2. I found the system to be simple. (As opposed to complex.)
3. I thought the system was easy to use.
4. I think that I could use the system without the support of a technical person.
5. I found the various functions in the system were well integrated.
6. I thought there was a lot of consistency in the system.
7. I would imagine that most people would learn to use the system very quickly.
8. I found the system very intuitive.
9. I felt very confident using the system.
10. I could use the system without having to learn anything new.

Table 6.1: Questionnaire

Usage statistics

Although user statistics are collected during the one day period the user learns to use the system, a choice was made not to analyze them, as they don't reflect how the user would use the system in a real environment that

well. In order to get representative statistics the prototype should be used during a longer period of time.

However, we are particularly interested to which extent users use the provided features. How many activities do they create, and how often do they switch between them? In order to be able to answer these questions to some extent, we ask specifically for them during the interview.

6.1.3 Task switch test

An overview of the second experiment is given in Table 6.2. The main influencing factor being analyzed is the interface used to perform the tasks. A within-subjects design is used where all subjects perform similar time boxed tasks twice; once using an unmodified Windows 7 environment and once using the prototype. A paired comparison is made between them. A within-subjects design is chosen since not all subjects are equally proficient in the tasks out of which the test is comprised. In order to lessen learning effects, the order of both tests is varied between users. The tasks under both conditions are similar, but the user is presented with a different data set to work on. Although precaution is taken in making the two separate data sets as similar as possible, they still might vary in difficulty. Therefore the used data sets are cross-balanced as well. In order for the test results to be fully balanced, multiples of 4 test subjects are needed.

The total test takes two hours. Location and computer used vary between subjects, but never between the two tested conditions. An initial 10 minutes are used to explain the procedure and tasks. In between both tests, 10 minutes are reserved to recuperate from the heavy workload. This leaves 50 minutes per test, during which work needs to be done on 4 separate tasks. The user is notified when to switch tasks at predetermined intervals, spreading the work in sessions of 2, 4.5 and 6 minutes, totaling in 12.5 minutes of work per task. Throughout the procedure the users are monitored, allowing to address any late responses to the notifications. The sequence of tasks A, B, C and D is depicted in Figure 6.1.

It can occur that a test needs to be interrupted due to an unclarity of the tasks or a bug in the software. When this happens the timer is paused until the issue is resolved.

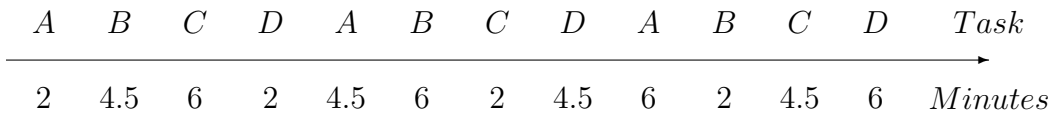

Figure 6.1: Task sequence

Goal	Study the effect of organizing work in semantically meaningful activities, managed in time and space.
Independent variable	Interface used: Traditional Windows 7 or Prototype
Dependent variables	Task Progress, Correctness, Task Load, Resumption Lag
Design	Within-subjects

Table 6.2: Experiment overview part 2

Tasks

The test will be comprised of general every-day tasks which any user can do; this to facilitate finding subjects. The progress of every task needs to be measurable and the progression when working on them should be fairly linear. A task isn't meant to be finished, so enough work needs to be available even for the most productive users. To prevent users from prioritizing quantity over quality the correctness of the work also needs to be verifiable.

Digitize text A scanned text needs to be hand-typed in a simple text editor. The text doesn't contain any formatting, so the amount of text the user is able to type is a direct indication of task progress. The amount of erroneous characters over the total amount of typed characters can be used as an error percentage. The scanned text is interspersed with written assignments for the participants, asking them to look up certain image resources on their disk drives. The text visible in those images needs to be typed into the text editor, substituting the assignment.

Searching Internet A text document contains a list of short calculations which need to be done. The numbers to use in those calculations originate from sources on the Internet which subjects have to search for themselves. The following is an example of one such calculation.

Height of the Eiffel Tower in meters + year when it was completed

Correctness is measured by assigning three points per calculation. One point for each source to look up, and a third point for the correctness of the calculation itself. In case there are different sources stating different numbers, any of them is considered to be correct. It is possible to use wrong numbers but do a correct calculation. The calculator application on the computer may be used.

Finding differences All the differences between two text documents need to be highlighted. The text needs to be processed linearly from the start. The position in the text of the last difference the user found is used as a progress measure. Any skipped differences over the total available differences up to that point make up the error.

Matching images The subjects are presented with an image which they need to recognize in between a set of other images. This set consists of several folders. Each folder contains images taken of one particular type of object. (e.g. bridges, islands, ...) First the subject needs to identify in which folder to look for the matching image, before trying to find it. When the image is found the subject writes down the filename in a text document and proceeds to the next image.

6.1.4 Study variables

The comparative study consists of one independent variable, *interface*. Either the prototype, or just the traditional Windows 7 environment is used. There are four dependent variables:

- **Task Progress:** Progress on the given tasks.
- **Correctness:** The correctness of the produced work during the tasks.
- **Task Load:** Overall experienced workload.
- **Resumption Lag:** How long it takes for users to pick up a task where they left off.

Task Progress, Correctness and Task Completion measure how well the subjects performed on the given tasks. Task Load is an indication of work load which is measured by using a modified version of the NASA Task Load Index (TLX) test, Raw TLX. Since we are particularly interested in the overhead of switching between activities, one additional measurement is done manually. Resumption Lag is the time between stopping work on a previous activity after having received the notification, and having opened all the required windows in order to continue work on the other activity. Automating this measurement under both conditions proved to be too difficult, and having the user do this manually would be too intrusive. The time taken to respond to the notification is monitored as well, and is not included into the Resumption Lag.

Performance measurements

Task Progress is measured per task, and represents the percentage of work the user was able to finish out of all available work on that specific task. Correctness is a percentage indicating how much work out of the processed work was done correctly.

Raw TLX (RTLX)

NASA TLX is a multi-dimensional rating procedure that provides an overall workload score based on a weighted average of ratings on six subscales. [21] An overview of the subscales is given in Table 6.3. RTLX is a common modification to NASA TLX which eliminates the weighting process by simply averaging or adding the ratings to create an estimate of overall workload. [22] Due to the variety of tasks we feel the user would have difficulties deciding on proper weights. Defining separate weights for each task would require too much time.

Prior to filling out the TLX test, the subjects are familiarized with the scales. The RTLX test is performed right after time is exceeded for each condition of the experiment; once after completing the tasks using the traditional Windows 7 interface, and once after using the prototype.

6.1.5 Hypotheses

Based on the research question posed in section 1.2, we formulate four null hypotheses. See Table 6.4.

$H1_0$	There is no significant difference in Task Progress between Windows 7 and Prototype.
$H2_0$	There is no significant difference in Correctness between Windows 7 and Prototype.
$H3_0$	There is no significant difference in Task Load between Windows 7 and Prototype.
$H4_0$	There is no significant difference in Resumption Lag between Windows 7 and Prototype.

Table 6.4: Experiment overview

$H1_0$ seeks to determine whether using the prototype has an effect on the amount of work which users can perform, while the second hypothesis ($H2_0$) verifies whether it has an effect on how well the work was done, measured as the percentage of work which was done correctly. $H3_0$ verifies whether

Mental Demand	How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy or demanding, simple or complex, exacting or forgiving?
Physical Demand	How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
Temporal Demand	How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
Performance	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
Effort	How hard did you have to work (mentally and physically) to accomplish your level of performance?
Frustration Level	How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

Table 6.3: NASA TLX subscales

using the prototype has an effect on perceived task load, as measured by the TLX test. Finally, $H4_0$ seeks to determine whether the interface influences the time needed to switch between different tasks.

6.2 Results

To test the hypotheses, Paired two-tailed Student t-tests are used with a chosen confidence level of 95%. The results aren't fully counterbalanced; one test where the prototype interface (Laevo) with task set A is tested first is missing.

Learning effects There was a significant learning effect between both parts on Task Progress ($N = 15$, $df = 28$, $p < 0.001$), but not on Correctness. On average participants performed 21% more work during the second part.

Figure 6.2: Progress learning effects

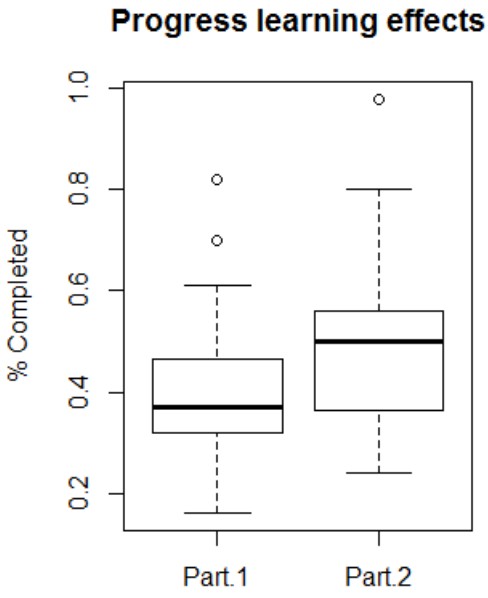

Figure 6.3: Progress learning effects

Task sets Two different task sets are used under both Interface conditions. These data sets are cross-balanced to compensate for possible differences in difficulty. There was no significant difference in either Task Progress (N = 15, df = 28, p = 0.97) or Correctness (p = 0.35) across both task sets.

Task Progress There is no significant influence from Interface used on Task Progress, thus the null hypothesis ($H1_0$) can't be rejected. (N = 15, df = 28, p = 0.59)

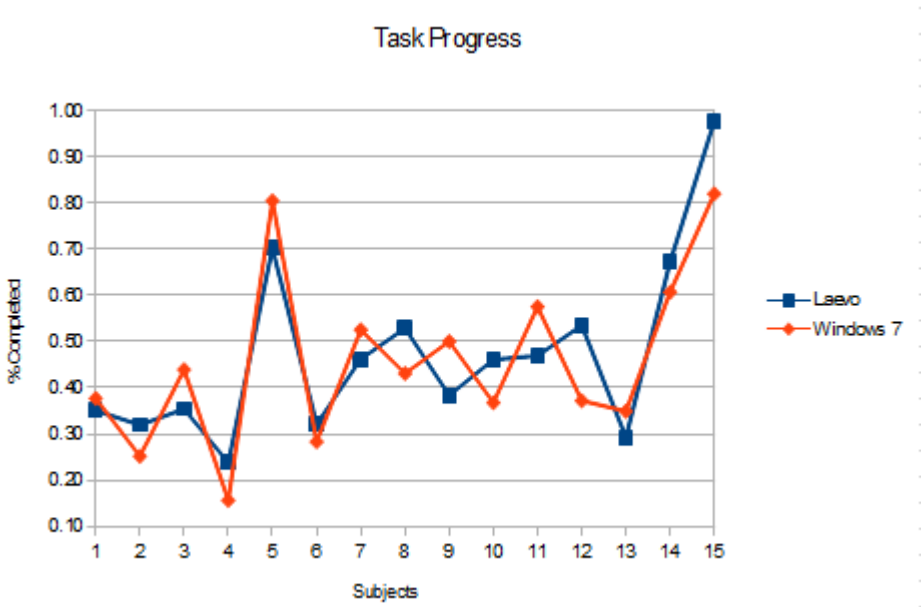

Figure 6.4: Task Progress measurements

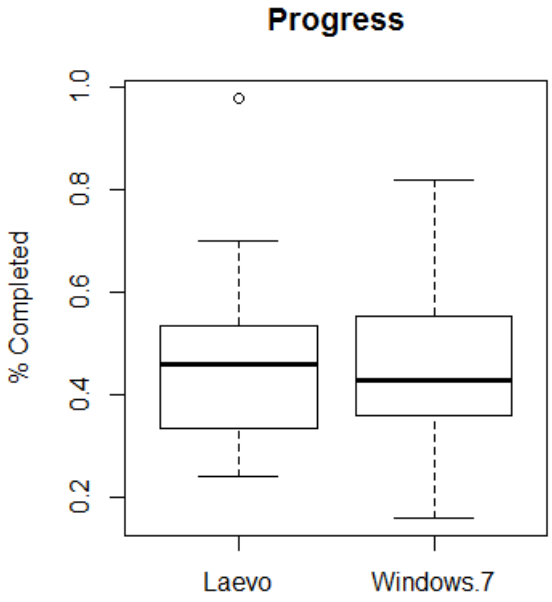

Figure 6.5: Descriptive statistics for Task Progress

Correctness The Interface used has a statistically significant impact on Correctness, thus the null hypothesis can be rejected. (N = 15, df = 28, p = 0.02) On average participants made 3% less mistakes when using the prototype interface.

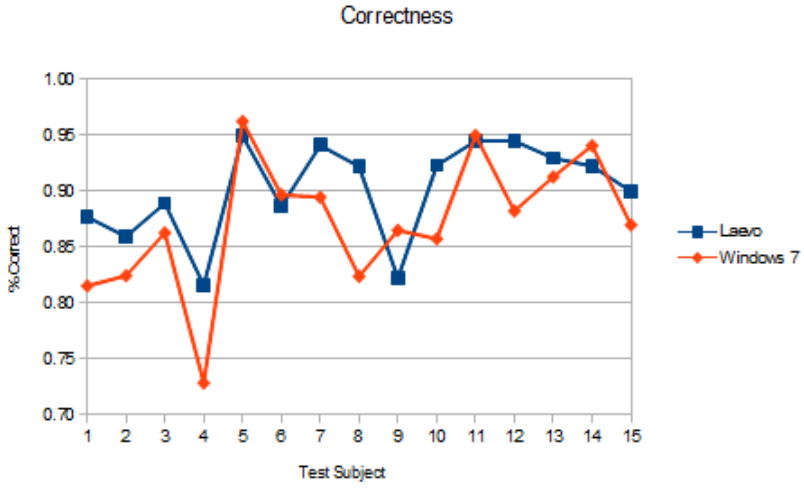

Figure 6.6: Correctness measurements

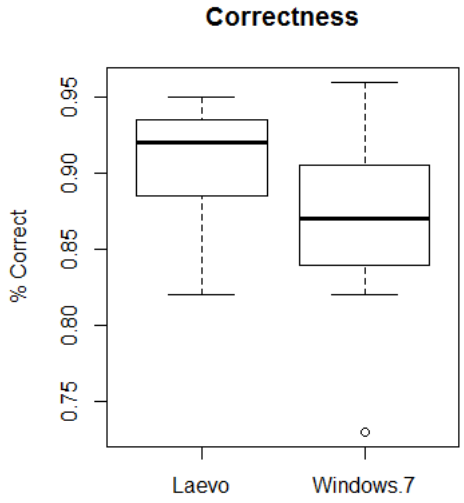

Figure 6.7: Descriptive statistics for Correctness

Task Load Three tests were omitted since there were major problems with the performed work. Task results for one participant were lost, and two other participants performed substandard. However, their TLX ratings are still considered to be valuable. The results aren't fully counterbalanced for the

order in which the interfaces were used. One extra participant started with traditional Windows 7 first.

The Interface used has a statistically significant impact on perceived Task Load, thus the null hypothesis can be rejected. ($N = 18$, $df = 34$, $p = 0.01$) On average participants rated the overall workload performed under the prototype interface 18% lower.

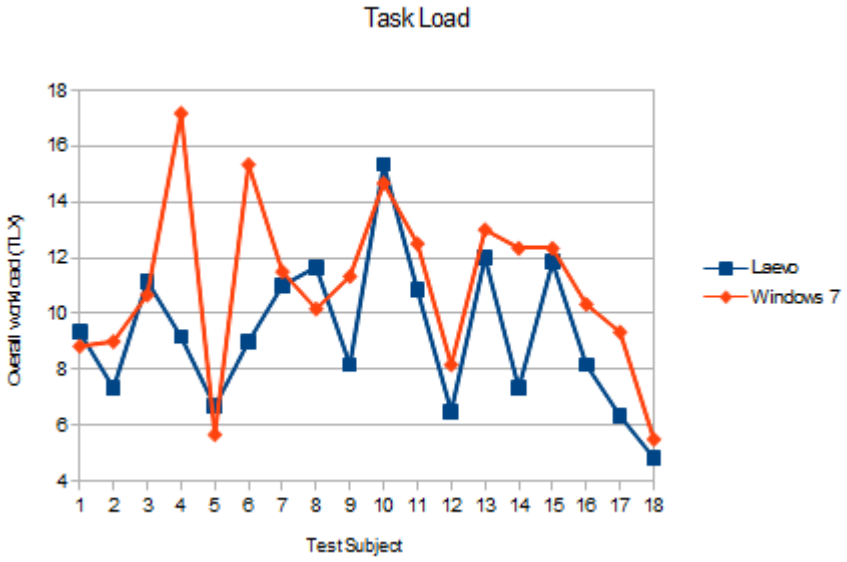

Figure 6.8: Task Load measurements

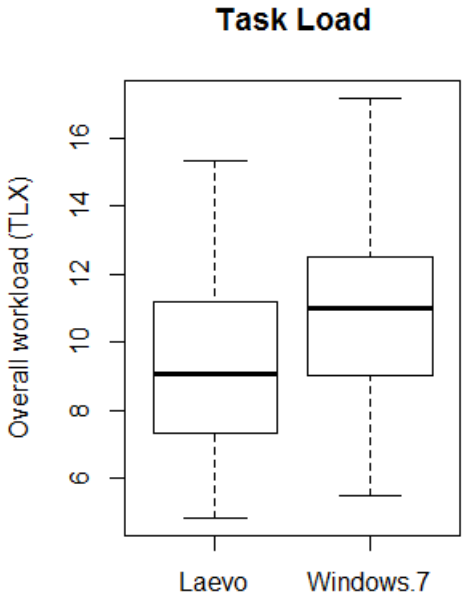

Figure 6.9: Descriptive statistics for Task Load

Resumption Lag We feel counter-balancing isn't of much importance when analyzing resumption times, so all measured results were originally taken into account. Graph 6.10 contains averages of all the measured results, except for two users with extremely long resumption times. ($>$ 2 minutes) Two outlier points from two separate users were removed as well. Only after those changes did the data follow a normal distribution.

The Interface used has no statistically significant impact on Resumption Lag, thus the null hypothesis can't be rejected. (N = 12, df = 22, p = 0.08)

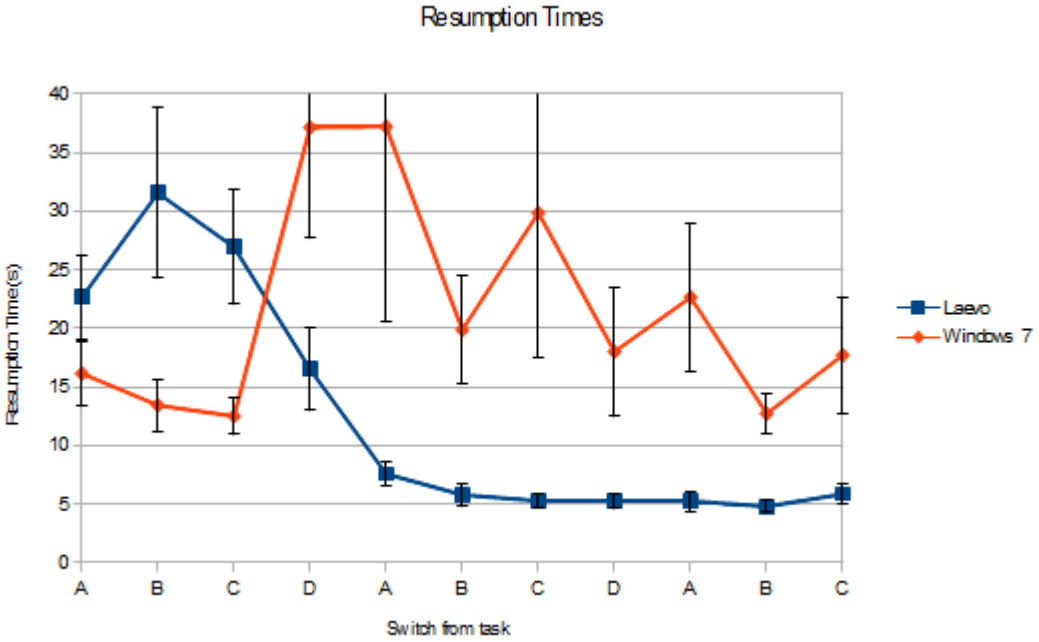

Figure 6.10: Resumption Lag

Since we noticed a lot of variance in the results due to differences in how well users are able to work with Windows 7, a distinction was made between experienced and inexperienced Windows 7 users. 6 users had an average resumption time < 15 seconds, which we classify as experienced. Another 6 had an average resumption time ≥ 15 seconds, which we classify as inexperienced.

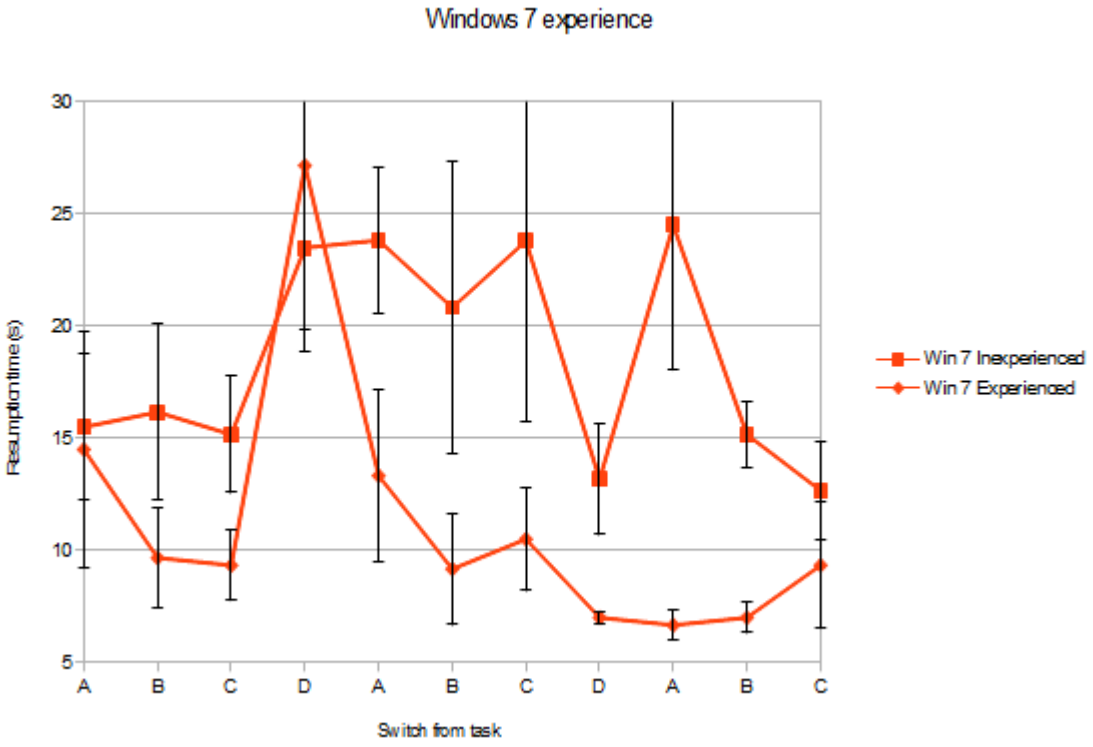

Figure 6.11: Experience under Windows 7

It is clear that the first time users need to switch to tasks, and thus need to construct their work environments, there is a lot more variance in the results. These four initial task switches will be dropped in the following analysis.

There is no significant influence from Interface used on Resumption Lag for experienced users, thus the null hypothesis can't be rejected. ($N = 6$, $df = 10$, $p = 0.08$) However, the Interface used has a statistically significant impact on Resumption Lag for inexperienced users, thus the null hypothesis can be rejected. ($N = 6$, $df = 10$, $p = 0.01$) On average it takes inexperienced Windows 7 users 2.7 times longer (12 seconds) to resume a task under the traditional Windows 7 environment than under the prototype interface.

Graph 6.12 highlights the difference between experienced and inexperienced users, and the overall effect the prototype interface has on resumption time.

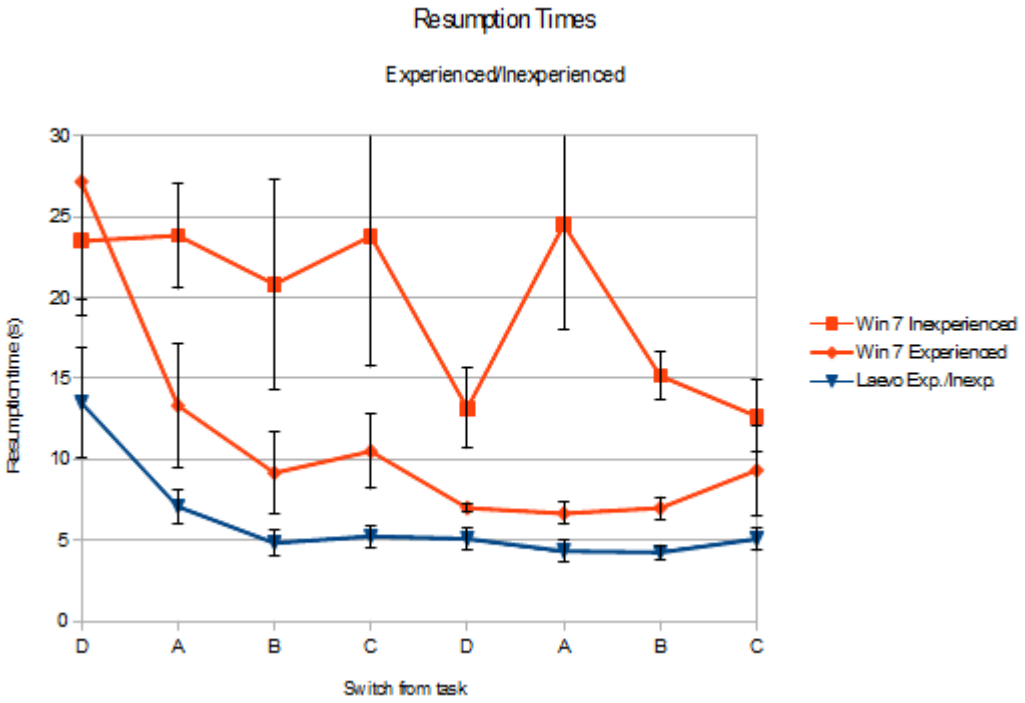

Figure 6.12: Resumption Lag Experience

6.3 Discussion

6.3.1 In-field study

SUS rating After having used the prototype for at least a full day, users filled out the System Usability Scale (SUS). Bangor et al. report an average SUS score of 76.2 for Graphical User Interfaces. [2] The average rating for the prototype interface is 83.5. (N = 12) Four people had prior experience with virtual desktops. They were asked to rate their favorite virtual desktop environment, resulting in an average rating of 76. (N = 4)

Although there are still many shortcomings in the UI, we can safely say the initial impression most users have is a positive one.

Tidy Many users valued the tidiness the prototype interface offers. When expressing how the interface improved their working environment words like “clear”, “organized”, “breathing space” and “clean” were used.

Usefulness All users, except one reported having to work on many activities simultaneously on a regular basis. Having to work on four simultaneous activities is deemed to be a realistic scenario. Some users mentioned the interface would probably be even more useful over longer time periods when the need arises to restart work on an old activity. One user continued using the interface during several weeks, and reported having the feeling she was able to get more work done. Many others refrained from continuing to use the interface due to several usability issues or bugs, but stated they would like to receive an update as soon as those have been addressed. Lastly, one user reported working "in" an activity reduces the temptation of switching to another "leisure" activity, as a sense of guilt arises from switching to it.

Ad-hoc activity description Although previous research has suggested that an ABC system should allow you to describe activities after having worked on them for a while, it seems to be equally important to easily and quickly allow to describe them right after creating them. This was requested multiple times. With the current interface you have to switch back to the time line in order to provide a name and icon.

Feature requests Many users expressed the desire of certain features, in line with what constitutes an ABC system. From this we can conclude that the interface brings across the concept of an ABC system well. Thinking in terms of activities seems intuitive, and even desirable after using the interface for a while. Some of the suggested features are:

- Allow to plan activities on the time line, including recurring activities.
- Allow to extract hours worked on activities.
- After closing an activity, allow to open it, restoring its state completely.
- Notifications originating from different activities should be hidden.
- Some applications need to be available across different activities.
- Provide a separate clipboard per activity.
- It would be nice to use an existing activity as a template of another, so you can create multiple instances of it and you no longer have to set up the initial work environment each time.
- You need to be able to search through your activities.

Compared to virtual desktops Two virtual desktop systems have been explicitly mentioned as less usable than our prototype interface.

Apple's "Spaces" is considered to be less useful by one person since depending on which application you open, you are automatically redirected to the workspace in which the application is located. The user prefers opening a new instance of the application, remaining within the work context of his current activity.

Two users mentioned recognizability issues with virtual desktops as implemented in Ubuntu. It only offers an outline of which windows are open within the desktop. They prefer our approach where you can specify a name and icon, allowing for immediate recognizability.

Alternative to calendar Interestingly, nobody expressed concerns that the interface is just another alternative for a calendar. When specifically asking for it, some people see a future for the interface as an entire replacement, or an alternate view for a calendar. Nobody found the provided functionality of the interface redundant.

6.3.2 Comparative study

First of all it should be noted that part 2 of the user study only measures a small portion of what constitutes an ABC interface. It measures the possible impact of grouping work in separate environments. A similar outcome can be achieved by using traditional virtual desktops. Therefore we can't draw final conclusions just from the reported results of the comparative study.

Task Progress No significant difference is found in Task Progress between Windows 7 and Prototype. If there would have been a small difference it would be unlikely to detect it due to the big learning effects on Task Progress. (See Graph 6.2)

Correctness A significant difference was found in Correctness between Windows 7 and Prototype. It is important to note that the Correctness measure has proven to be highly sensitive to outliers. If very little work is done on a particular task this can weigh in heavily on the overall correctness. Though a probability of 0.02 is reported, this drops to $p = 0.39$ when adding the two highly problematic test which were omitted. One of those users was only able to finish 4% of work compared to the best user on a particular task, and thus no representative correctness could be calculated. He didn't fit the description of a regular Windows 7 user. The other user seemed to have

great problems on the task where two English texts needed to be compared; he specifically mentioned not being good in English.

It would be premature to conclude that the prototype has a positive effect on Correctness. However, it is possible that decreased correctness indicates an increase in cognitive load as e.g. noted by Ayres. [1] In future studies great care should be taken in sensitivity and granularity of the correctness measure.

Task Load A significant difference is found in subjectively rated Task Load between Windows 7 and Prototype. It is worth noting the scale which contributes most to this difference is Effort. This is how hard the participants felt they had to work (mentally and physically) to accomplish their level of performance. An overview of probability values for the separate scales is given in Table 6.5. (N = 18)

Scale	**Probability**
Mental	0.09
Physical	0.06
Temporal	0.05
Performance	0.6
Effort	0.01
Frustration	0.06

Table 6.5: Probability values for the separate TLX scales

The Performance scale reflects that there is no significant difference in Task Progress between both interfaces. Users are significantly less rushed (Temporal) when using the prototype, while simultaneously feeling less Effort is needed to achieve their performance.

Resumption Lag Given the results as shown in Table 6.12 we feel it is safe to conclude that inexperienced Windows 7 users are slower in switching between tasks than experienced users. All users can switch between tasks faster using the prototype, although inexperienced Windows 7 users benefit the most. For inexperienced users the increased initial work for setting up the task environment is almost won back immediately when switching to the task again for the first time.

Observations While performing the tasks using the prototype interface, not all users felt the need to specify a name or icon to each activity. This way they managed to save time. This explains the high variance in the

initial setup time of the four first task switches. Regardless of whether users provided a name for the activities, they were able to identify which activity was which thanks to the temporal indication and/or arrangement in space. E.g. when asked to switch to activity C they recalled this was the third activity, and simply opened the third visible activity on the time line.

Chapter 7

Conclusions

The main contribution of this thesis is to explore ways in which activities can be managed more easily within an ABC system. The main research question brought forward is:

> *Can organizing activities in time and space simplify managing them and reduce information overload?*

Our approach of integrating a calender with the desktop interface by implementing activities as separate virtual desktops represented on a time line was extremely well received during user studies. People see a lot of advantages such an interface could offer, and described additional possibilities in detail. The prototype did not only succeed in conveying how activities could be managed over longer periods of time, but additionally also managed to clearly bring across the concept of ABC. Even during the short comparative user study visualizing activities in time helped users in distinguishing between activities. Integrating a calendar into an ABC system is a worthwhile path to continue exploring in further research.

During a comparative user study, work which needed to be done on four concurrent activities was rated to be easier when Windows 7 was augmented with the prototype interface, allowing to group work in separate virtual desktop environments. Feedback from an in-field user study gives some extra insight into this result; people like working in the "clean" workspaces offered by virtual desktops. The prototype allows faster switching between concurrent activities, especially for less experienced Windows 7 users. Additionally there is an indication that less errors are made when using the prototype interface, which could hint at a reduced cognitive load when using dedicated project spaces. We therefore propose a design recommendation of implementing activities as dedicated project spaces, or more specifically virtual desktops in

an ABC system. Using a Personal Information Management (PIM) approach wouldn't provide the outlined advantages.

As a second research question we asked: *"Can centralized handling of application notifications help in classifying activities and handling interruptions?"* Unfortunately our eventual prototype did not implement any of the described features which would allow us to verify the merit of such an approach. However, during the in-field user study several users brought forward issues with notifications popping up (e.g. email notifications) originating from applications open in other activities. Others requested the ability to have some applications be available in all activities in order to resolve this issue. (e.g. the email client) This feedback hints at the need to better support notifications in an ABC system.

Appendix A

Laevo User Manual

Thank you for showing an interest in Laevo and wanting to participate in this user study. The following manual will explain you all the features of Laevo, and how to use them to enrich your daily work. As this is still a prototype the feature set is limited, and it shouldn't take you too long to try them all out. Bugs can occur, and some are known. When you encounter one, be sure to check out Section A.5.

A.1 What is Laevo?

Laevo is a novel user interface prototype intended to help you organize work in terms of much larger and thematically connected units of work than the traditional Windows 7 interface allows you to. These units of work are called "Activities". It isn't difficult to imagine what an activity looks like when working on it. It is visually indistinguishable from your ordinary work environment, the desktop. What is different is you have not one, but several desktop environments at your disposal. This concept is often called virtual desktops. Laevo attempts to broaden on the concept of virtual desktops by incorporating concepts of Activity-Based Computing (ABC). ABC has emerged as a response to the traditional application- and file-centered computing paradigm, which is oblivious to a notion of a user task spanning heterogeneous devices, multiple applications, services, and information sources. (http://activitybasedcomputing.org/) Laevo aims to one day become a full-fledged ABC system, but for now mainly acts as a virtual desktop environment. This doesn't prevent it from being able to contribute to existing ABC research, as all the following described features were designed to contribute to open research questions raised by earlier prototypes.

Activities *So what is an activity?* When working on an activity, it looks just like your normal desktop environment. You can have several application windows open, start up new applications or browse for files. The only difference is these open windows are "assigned" to the activity you are working on. They will only be visible within this activity. When switching to another activity only the windows assigned to that activity will be visible. This allows you to group semantically relevant windows together, and hide those that are unimportant for the task at hand.

Activity states An activity can either be *open*, or *closed*. Several activities can be open simultaneously, but you can only work on one activity at a time. The activity you are working on is called the *active* activity. An activity stays open just as long as you want it to. It is up to the user to close activities which are no longer useful.

A.2 Starting and exiting Laevo

As a prerequisite for running Laevo you need to have .NET 4.0 installed. This is a recommended update on Windows Update, which by default is installed automatically.

Laevo is a simple executable, it doesn't need to be installed. The zip file provided to you contains a folder with the application, unzip it to wherever you like. You can start the application simply by double clicking the "Laevo" executable.

The first time you start Laevo an activity is automatically created for you. All the windows you had open prior to starting Laevo will now be assigned to this "Startup" activity. This activity is automatically opened, so you won't notice much of a difference with the way your desktop looked like prior to launching the application. The only indication you have of Laevo running is a tray icon in the bottom right corner of your screen, as depicted below.

Figure A.1: Laevo tray icon

By default Windows hides tray icons when there are too many. You can

show all tray icons by clicking on the left-most arrow icon. It is recommended to change the visibility setting for the Laevo icon to always visible. This is possible by clicking the arrow icon, selecting "Customize", identifying the Laevo icon and setting its behavior to "Show icon and notifications".

Figure A.2: Hidden tray icons

Tray icon menu By right clicking on the Laevo tray icon, a list of options appear. Through here you can exit the application, after which your desktop environment will be restored as if nothing happened. The other functions in this menu will be explained in the following sections. We will refer to this menu as the *"tray icon menu"*.

Restarting Laevo Upon starting Laevo after having used it before, a message will pop up.

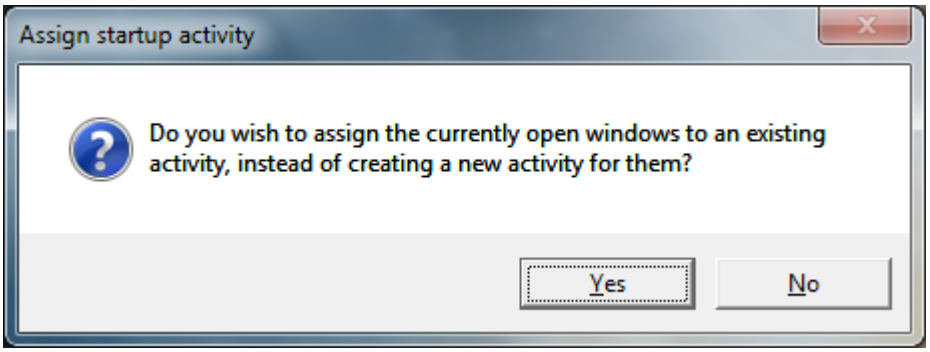

Figure A.3: Startup message box

In case you click "Yes", the time line which shows you an overview of all your activities (see Section A.3) will pop up, and you are expected to find a previously created activity on which you want to continue working. All the currently open windows will be assigned to the selected activity. You select an activity by clicking on its rectangular area. It is possible no activity is visible initially, and you have to zoom out and/or go back in time in order to see the older activities.

When you click "No", the same will happen as when you started Laevo for the first time. A new "Startup" activity is automatically created for

you, and opened. The currently open windows will be assigned to this new activity.

A.3 The time line

The time line offers you a full screen overview of your activities. You can access it at any time by pressing the "Caps Lock" key. The default behavior of Caps Lock is overridden. You can still turn Caps Lock on or off using the key combination "Caps Lock - A", or by accessing the "Turn Caps Lock On/Off" option in the tray icon menu. The horizontal axis represents time, while the vertical axis can be used to lay out activities as you wish. A yellow marker highlights the current time on the time line. An activity is represented on the time line by a colored rectangular area. Horizontally, it occupies the space during which the activity was open. All open activities will continue expanding up to this marker for as long as they are open. A closed activity can be recognized after some time as a gap appears in between the time it was closed and the current time marker.

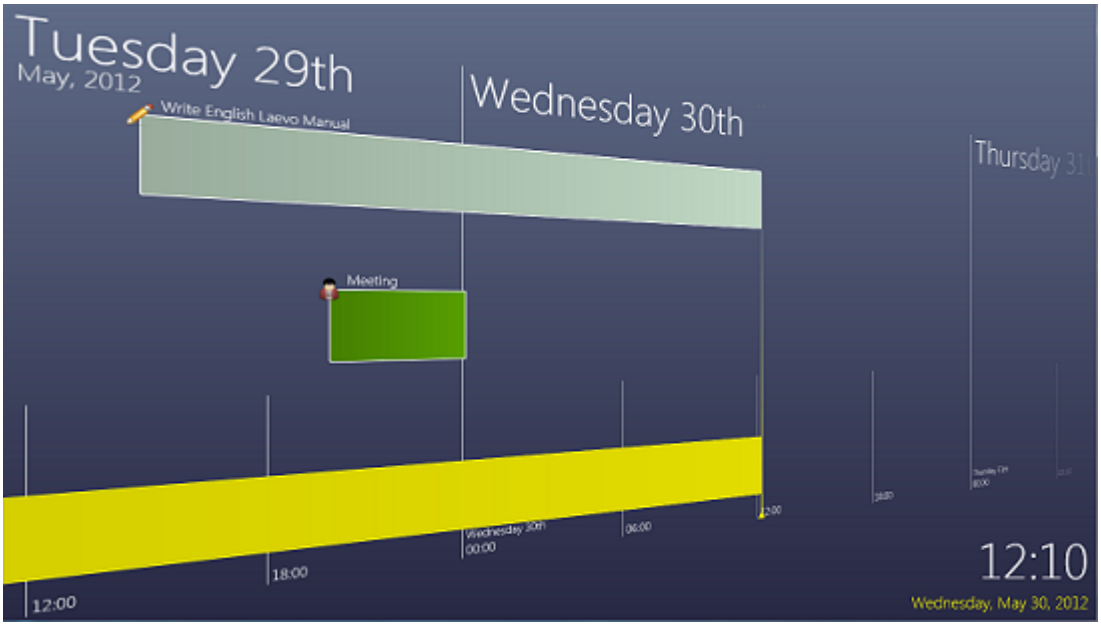

Figure A.4: Activity time line

You can access any point in time on the time line by using two operations.

- Move the time line backwards/forwards in time by clicking and dragging horizontally.

- Zooming in/out using the scroll wheel. The position where your mouse is located when scrolling is the position around which is zoomed in/out.

After some time you will be able to make out yellow lines which are shown within the rectangular areas of the activities, called *attention lines*. These indicate when the activity was active, meaning it was the activity you were working on at that time.

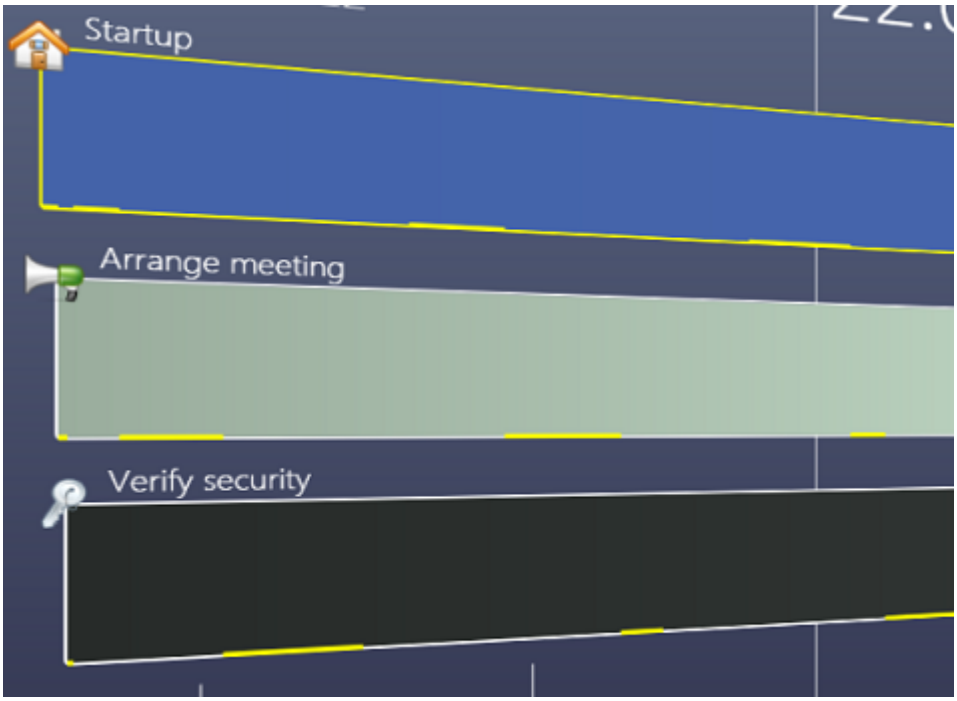

Figure A.5: Attention lines

Managing activities You can relocate activities vertically by clicking and dragging the colored area. In order to help you recognize activities better you can assign a name, icon and color to them. All of these can be specified in the edit activity pop-up which is opened by clicking the activity icon.

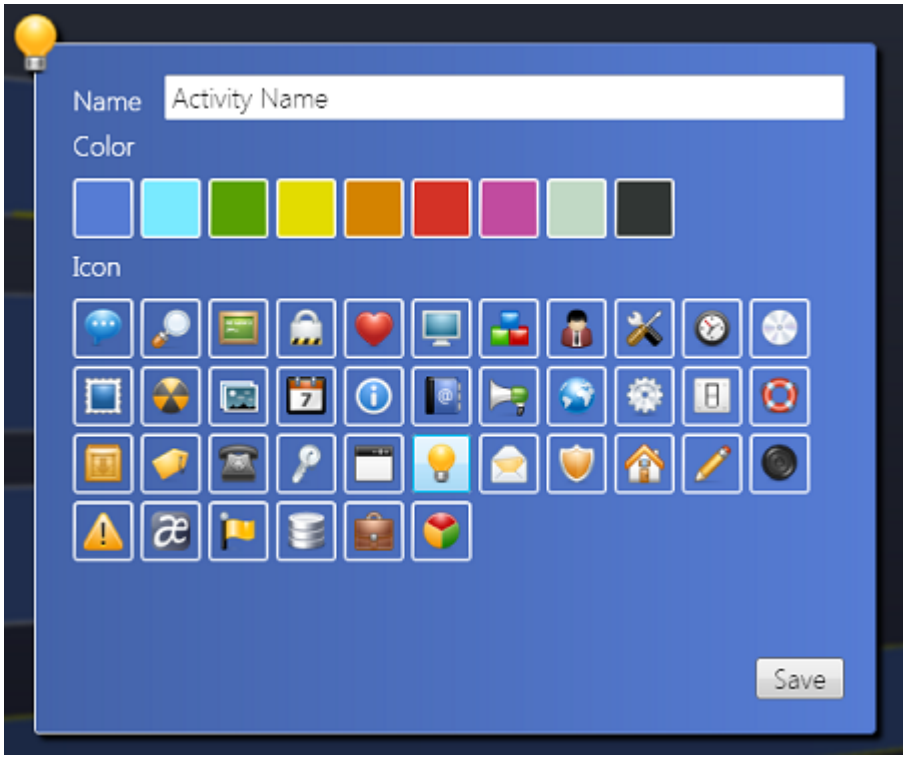

Figure A.6: Edit activity pop-up

You can also adjust the activity name directly from the time line by clicking on it. An input caret will appear and you will be able to start typing as usual. When you are finished editing the name, press “Enter” or “Escape”.

Switching between activities Whenever you want to switch between activities the time line is the place to be. Press “Caps Lock” and it will pop up. The activity which you were working on previously has a yellow border, while the others have a white border. You can switch to an activity by clicking on it’s colored area. This will take you to the desktop environment for that activity. You can also re-open a previously closed activity by clicking on it. Afterwards it will show up on the time line as if it was never closed.

A.4 Activity desktop

For each activity you create you get a dedicated desktop environment. It looks exactly as your ordinary desktop environment, except for the Laevo tray icon which is visible in the bottom right corner. You have a few additional

activity-specific options available which are explained in this section. They are either available through the tray icon menu, or by using shortcut keys.

New activity At any point (even from the time line) you can create a new activity. An empty desktop environment is created for it and immediately opened. You can access the "New Activity" option through the tray icon menu or by using the shortcut key "Caps Lock - N". You can check out your new activity on the time line. From there you can edit its name, icon and color.

Closing activity Once you finish work on an activity, you might want to close it so it no longer occupies space on the time line. By closing it it will no longer expand up to the current time marker. It is important to mention that the "Startup" activity can never be closed. This is the first activity you select when starting up Laevo. Therefore it is recommended to use this activity as a place where you can perform work which isn't directly related to any other activity.

You can access the "Close Activity" option through the tray icon menu or by using the shortcut key "Caps Lock - W". This option is disabled as long as there are any windows open within the activity. After closing an activity, the time line pops up. You will be unable to exit the time line by pressing Caps Lock, as the activity you came from is now closed. You are expected to open an existing activity to continue work on.

Activity library Each activity has a dedicated library to store files in. You can access it through the tray icon menu by selecting "Open Current Activity Context" or by using the shortcut key "Caps Lock - L". Additionally you can also always find it under "Libraries" in Windows Explorer, under the name "Activity Context".

At any given time, the activity library will only show the files of the currently open activity. Accessing files of another activity requires you to switch to it first.

This library is an ordinary Windows 7 library and thus also provides you with all the respective functionalities. Most importantly, you can include new folders of which files should show up in the library. This allows you to maintain a traditional way of organizing your files, but easily having them at hand when working on the activity.

Cutting and pasting windows Sometimes it might be useful to move a window from one activity to another. There is no graphical user interface

available to do this. However, there are two easy to remember shortcut keys which might seem familiar if you are used to cutting and pasting using the "Ctrl - X" and "Ctrl - V" shortcut keys. In order to cut and paste a window respectively you can use "Caps Lock - X" and "Caps Lock - V". The window which got focus last is cut. You can ensure the desired window has focus by activating it, e.g. by clicking on it. After cutting a window it will immediately be hidden.

You can cut multiple windows in a row. When pasting, all the cut windows will become visible again. By cutting windows in one activity, and pasting them in another, you can move windows between activities.

A.5 Yikes! A bug.

This is still a prototype, meaning some bugs are to be expected. It's easy to work around the most common ones. Unless the entire application crashes you can usually still recover by following the steps outlined below.

Shortcut keys stop working Sometimes the shortcut keys stop working. When this occurs pressing Caps Lock will result in its usual behavior, turning Caps Lock on or off. Verify whether the program is still running by opening the tray icon menu. If the menu pops up the program didn't crash. Try Caps Lock once more. If it still doesn't work follow the guidelines in *When all else fails* below.

White screen When the time line pops up as a white screen, the user interface has crashed. You can reset it by going to the logon screen and back. E.g. by using the key combination Ctrl-Alt-Delete.

Shadows stay behind From time to time, shadows of certain windows can stay behind. For now no time was invested in fixing this as it is only a minor issue. If you are annoyed by them too much you can restart Leavo by following the guidelines in *When all else fails*.

Performance issues On older PCs it might be possible the time line doesn't scroll smoothly. You can attempt lowering the quality of the time line in the settings which can be accessed from the tray icon menu.

When the time line only degrades in performance after having used it for a longer period, try disabling the "attention span lines" in settings. Currently they aren't optimized and can have a big impact on performance.

When all else fails When all else fails, but the program didn't crash, you can still restart Laevo. A clean exit will ensure that all edits to the existing activities are saved. All the windows of open activities will become visible again. After restarting Laevo you will have to restore the environment to its desired state manually. Upon startup, choose "Yes" and select your previous startup activity. All the open windows are now assigned to this activity. By cutting and pasting windows, you can move them to the correct activities again.

Crash When the application crashes, Windows normally notifies you of this. If all goes well, there should be a "log" text file in the same folder than the Laevo executable. This log file contains valuable information for me so I can prevent any crashes in the future. You can help me out by sending it to me.

A.6 Removing Laevo

Had enough of all these virtual desktops or you find those pesky bugs too annoying to continue using Laevo for now? This section will shortly outline the steps you need to take to remove Laevo entirely, without losing any possible important data.

Removing the library Laevo creates a Windows library called "Activity Context" which can be found in the left side bar of Windows Explorer under "Libraries". You can easily remove it by right-clicking it and selecting "Delete".

Removing data Your activities and its associated files are stored under "Documents\Laevo". In case you stored files in the activity context library, they will be located in the subfolder "Activities". Each activity has a separate folder, in which the files for that activity are located. You might want to copy those files you still want to keep. You can simply remove the remaining "Laevo" folder.

Appendix B

Glossary

activity context is the entire relevant context of a subject in a certain activity. This includes all the relevant resources, tools, contacts, ... as shown in Engeström's Activity System Model (see Figure 3.2). 20, 21

dedicated project spaces are spatially defined subsets of a virtual work environment that provide contexts for individual projects or types of tasks [25]. 10, 28

episodic memory is the memory of autobiographical events (times, places, associated emotions, and other contextual knowledge) that can be explicitly stated. 28

externalization is the transformation of internal mental activities into external ones. 22

information overload is the difficulty a person can have when having to deal with too much information. 1

virtual desktops are user interfaces which expand the physical limits of the screen's real estate through the use of software. This compensates for a limited desktop area and can also be helpful in reducing clutter. 47

Appendix C

Acronyms

ABC Activity-Based Computing

AT activity theory

ACM Association for Computing Machinery

HCI human-computer interaction

HF human factors

OS operating system

PIM Personal Information Management

RTLX Raw TLX

SUS System Usability Scale

TLX Task Load Index

UI user interface

Appendix D

Bibliography

[1] Paul Ayres. Impact of reducing intrinsic cognitive load on learning in a mathematical domain. *Applied Cognitive Psychology*, 20(3):287–298, 2006.

[2] A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[3] Liam Bannon. Design at work. chapter From human factors to human actors: the role of psychology and human-computer interaction studies in system design, pages 25–44. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1992.

[4] Liam Bannon, Allen Cypher, Steven Greenspan, and Melissa L. Monty. Evaluation and analysis of users' activity organization. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '83, pages 54–57, New York, NY, USA, 1983. ACM.

[5] Jakob Bardram, Jonathan Bunde-Pedersen, and Mads Soegaard. Support for activity-based computing in a personal computing operating system. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 211–220, New York, NY, USA, 2006. ACM.

[6] Jakob E. Bardram. Activity-based computing - lessons learned and open issues, 2004.

[7] BeeDocs. Beedocs timeline 3d. `http://www.beedocs.com/timeline3D/`.

[8] Steven Brint. Gemeinschaft revisited: A critique and reconstruction of the community concept. *Sociological Theory*, 19(1):123, 2001.

[9] J. Brooke. SUS: A quick and dirty usability scale. In P. W. Jordan, B. Weerdmeester, A. Thomas, and I. L. Mclelland, editors, *Usability evaluation in industry*. Taylor and Francis, London, 1996.

[10] Stuart K. Card and Austin Henderson, Jr. A multiple, virtual-workspace interface to support user task switching. In *Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface*, CHI '87, pages 53–59, New York, NY, USA, 1987. ACM.

[11] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.*, 41(1):2:1–2:31, January 2009.

[12] Mary Czerwinski, Eric Horvitz, and Susan Wilhite. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 175–182, New York, NY, USA, 2004. ACM.

[13] Yrj Engestrm. Activity theory and individual and social transformation. In Yrj Engestrm, Reijo Miettinen, Raija-Leena Punamki, and Raija-Leena Punamki-Gitai, editors, *Perspectives on activity theory*, pages 19–38. Cambridge University Press, Cambridge, UK, 1999.

[14] Scott Fertig, Eric Freeman, and David Gelernter. Lifestreams: an alternative to the desktop metaphor. In *Conference companion on Human factors in computing systems: common ground*, CHI '96, pages 410–411, New York, NY, USA, 1996. ACM.

[15] Leah Findlater and Joanna McGrenere. A comparison of static, adaptive, and adaptable menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 89–96, New York, NY, USA, 2004. ACM.

[16] Center for Activity Theory and Developmental Work Research. The activity system. `http://www.edu.helsinki.fi/activity/pages/chatanddwr/activitysystem/`.

[17] Harold Garfinkel. *Studies in ethnomethodology*. Prentice-Hall, 1967.

[18] Victor M. González and Gloria Mark. "constant, constant, multi-tasking craziness": managing multiple working spheres. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 113–120, New York, NY, USA, 2004. ACM.

[19] Elizabeth Goodman, Erik Stolterman, and Ron Wakkary. Understanding interaction design practices. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 1061–1070, New York, NY, USA, 2011. ACM.

[20] Google. Google chromebook. `http://www.google.com/chromebook`, 2011.

[21] S. G. Hart and L. E. Stavenland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati, editors, *Human Mental Workload*, chapter 7, pages 139–183. Elsevier, 1988.

[22] Sandra G. Hart. Nasa-Task Load Index (Nasa-TLX); 20 Years Later. In *Human Factors and Ergonomics Society Annual Meeting*, volume 50, 2006.

[23] Inc. Humanized. Humanized enso. `http://humanized.com/enso/`, 2008.

[24] Susanne Jul and G.W. Furnas. Critical zones in desert fog: Aids to multiscale navigation, 1998.

[25] Victor Kaptelinin and Bonnie A. Nardi. *Acting with Technology: Activity Theory and Interaction Design*. The MIT Press, 2009.

[26] Aparna Krishnan and Steve Jones. TimeSpace: activity-based temporal visualisation of personal information spaces. *Personal & Ubiquitous Computing*, 9(1), 2005.

[27] Kari Kuutti. *Activity theory as a potential framework for human-computer interaction research*, pages 17–44. Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.

[28] Mik Lamming and Mike Flynn. "forget-me-not" intimate computing in support of human memory, 1994.

[29] Kevin Larson, Maarten van Dantzich, Mary Czerwinski, and George Robertson. Text in 3d: some legibility results. In *CHI '00 extended*

abstracts on Human factors in computing systems, CHI EA '00, pages 145–146, New York, NY, USA, 2000. ACM.

[30] Aleksei N. Leontiev. The Problem of Activity in Psychology. *Soviet Psychology*, 13(2):4–33, 1974.

[31] Aleksei N. Leontiev. *Activity, consciousness and personality*. Prentice Hall, Englewood Cliffs, NJ, 1978.

[32] D. A. Norman. *The invisible computer : why good products can fail, the personal computer is so complex, and information appliances are the solution*. 1. mit press paperback ed edition, 1999.

[33] Rafael Nunez, Benjamin Motz, and Ursina Teuscher. *Metaphor and Symbol*, 21(3):133–146, 2006.

[34] Tim F. Paymans, Jasper Lindenberg, and Mark Neerincx. Usability trade-offs for adaptive user interfaces: ease of use and learnability. In *Proceedings of the 9th international conference on Intelligent user interfaces*, IUI '04, pages 301–303, New York, NY, USA, 2004. ACM.

[35] Jef Raskin. *The humane interface: new directions for designing interactive systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

[36] Jeff Sauro and James R. Lewis. When designing usability questionnaires, does it hurt to be positive? In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 2215–2224, New York, NY, USA, 2011. ACM.

[37] Lucille Alice. Suchman. *Plans and situated actions : the problem of human-machine communication / Lucy A. Suchman*. Cambridge University Press, Cambridge [Cambridgeshire] ; New York :, 1987.

[38] Phil Turner and Susan Turner. From description to requirements: an activity theoretic perspective. In *Ref. No ISO*, pages 286–295. ACM Press, 1999.

[39] Stephen Voida, Elizabeth D. Mynatt, and W. Keith Edwards. Reframing the desktop interface around the activities of knowledge work. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 211–220, New York, NY, USA, 2008. ACM.

[40] Christopher D. Wickens, John Lee, Yili D. Liu, and Sallie Gordon-Becker. *Introduction to Human Factors Engineering (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.