Laevo: A Temporal Desktop Interface for Integrated Knowledge Work

Anonymized
Anonymized
Anonymized
Anonymized

ABSTRACT

Prior studies show that knowledge work is characterized by highly interlinked practices, including *task, file and window management*. However, existing personal information management tools primarily focus on a limited subset of knowledge work, forcing users to perform additional manual configuration work to integrate the different tools they use. In order to increase *tool integration*, we review literature on how users' activities are created and evolve over time as part of knowledge worker practices. From this we derive the *activity life cycle*, a model describing the different states and transitions of an activity. The life cycle is used to inform the design of Laevo, a *temporal activity-centric* desktop interface for personal knowledge work. Laevo allows users to structure work within dedicated workspaces, managed on a timeline. Through a *centralized notification system* which doubles as a to-do list, incoming interruptions can be handled. Two field studies confirm the benefits observed during prior studies in activity-centric computing. Users appropriated the newly introduced time-based features in many different ways, incorporating them into their existing work practices.

Author Keywords

Personal information management; knowledge work; tool integration; activity-centric computing

ACM Classification Keywords

H.5.2 Information interfaces and presentation (e.g., HCI): User Interfaces – Graphical user interfaces (GUI)

INTRODUCTION

Knowledge work includes producing, transforming, consuming and communicating large amounts of diverse information. Traditionally, the digital artifacts associated with this work are managed using different personal information management (PIM) tools, each specializing in a particular task. Examples include calendars, to-do lists, email clients, file managers and web browsers. While a wide range of specialized tools allows users to select those most suited for their work, it also introduces *information fragmentation*: for each tool information items are stored, managed and viewed within a separate application-specific collection [6]. Studies have shown that information is often organized within hierarchies reflecting the users' projects or tasks [6, 7, 14]. Because a single project can make use of several tools, it is left to the user to associate the different related information items and maintain consistency across the separate project hierarchies. Additionally, many information items need to be passed from one tool to another, which is often still a manual process. For example email has been shown to be the primary source of new events entered in a calendar, which in turn is not only used for planning, but also for reporting purposes and recording memorable events [33]; folder hierarchies can act as kind of project plans [18]; action-oriented items like emails or temporary files are used as reminders [35].

Figure 1. Laevo is an *activity-centric* desktop interface that supports (i) persisting, restoring and managing window configurations and files within *dedicated workspaces* which are managed on timeline; (ii) handling *interruptions* and to-do items through a centralized notification system, in place or by redirecting them to new or existing workspaces.

Prior research has proposed the concept of activity as a structuring mechanism, organizing information items [4, 27, 34], communication and collaboration [16, 26] within the context of actual higher-level knowledge work. Existing *activity-centric* systems however, primarily focus on the ad hoc construction of activities, often neglecting the importance of other knowledge worker practices integral to their work, such as interruption handling, archiving and planning. Improving *tool integration* could overcome the significant overhead of manually having to configure and exchange information items.

Submitted for review.

Task, window and file management, usually supported by separate tools, occur at different points in time throughout the life cycle of an activity. In order to analyze the emergence and evolution of activities over time we review prior research. Based on this review we derive a temporal model for knowledge work, *the activity life cycle*. It depicts the different states and transitions of an activity over time. Informed by this model, we designed and implemented *Laevo*, a *temporal activity-centric* desktop interface specifically designed for knowledge work on a single personal computer within an office setting. It is capable of persisting, restoring and managing window configurations and files within dedicated workspaces which are managed on a timeline and a to-do list. The to-do list doubles as a *centralized notification system* through which incoming interruptions, like emails, can be handled, in place or by redirecting them to new or existing workspaces.

In this paper we (i) introduce a temporal model for knowledge work, *the activity life cycle*, (ii) describe the design and implementation of *Laevo* and (iii) report on two *field studies* during which participants used the system on their personal computers; one full day study with 12 participants, and a two week study with 6 participants.

LITERATURE REVIEW

Task, window and file management are an integral part of knowledge work. Widespread PIM tools primarily focus on supporting the management of only one of these. As a result, the user is required to coordinate tool use and pass information around. Prior research has explored tool integration to some extent, but often neglects at least one part of knowledge work. In this section, we review literature from different perspectives to gain insight in the commonalities and conflicts between them. By relating them to existing *activity-centric* computing systems we describe opportunities to further improve tool integration. An overview is given in Figure 2.

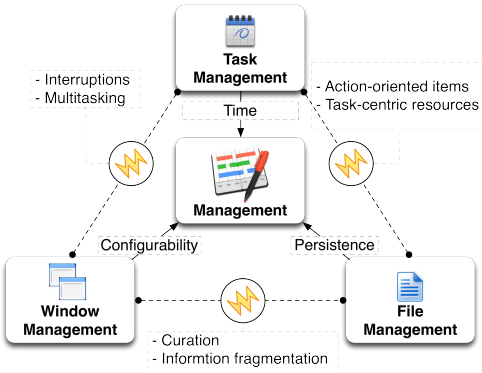

Figure 2. Overview of task, window and file management, highlighting conflicts between them. Through activity-centric computing, the separate tools can be better integrated.

Knowledge work practices

Window Management

In essence, window managers are tools that allow users to handle an increasing number of windows by allowing them to be spatially grouped together, thus leveraging human spatial memory. Switching between these separate window groups facilitates *multitasking*. Many modern window managers (such as those of Windows XP up to Windows 8) automatically support *application grouping* in which windows from the same application are grouped together under one item on the taskbar. Extending on this idea, GroupBar [31] gives the user more control by allowing them group arbitrary windows manually. Overall, existing window managers focus on similar lightweight *task management*. The Rooms system [15] was one of the earliest to provide such functionality, using *virtual desktops*. Work is distributed across several different desktop environments, each of which were *dedicated workspaces* only showing the windows open on that particular desktop. More recent window managers have explored alternate visualizations. One such example is Scalable Fabric [29], in which windows shrink in size when moving them into the periphery, allowing to organize them in labeled groups. WindowScape [32] offers implicit task management by storing snapshots of short-lived window groupings which can be returned to at later moment in time. Elastic Windows [20] allows organizing windows *hierarchically* and window operations can be performed on window groups, e.g. collapsing of entire window hierarchies. Spatial arrangement doesn't need to be restricted to 2D space, e.g. BumpTop [2] allows dragging and tossing around objects in 3D space, taking the desktop metaphor even further. However, long term task management like planning and reporting is not supported in any of these systems.

Task Management

We define a *task* as a set of actions which need to be performed in order to achieve a certain goal. In a desktop environment, one task is usually associated with several applications and documents. *Window management* helps out in organizing these during an ongoing multitasking session, but window configurations are lost once work is discontinued. In order to more easily re-initialize a working context users often aggregate related documents in task-specific folders [6, 7, 14]. Several systems explore integrating *task-centric resources* within existing applications. Taskmaster [5], e.g., recasts email as task management. Task Gallery [30] allows users to organize windows and files in tasks which are represented in a 3D desktop environment. Mylyn [22] is an extension for the Eclipse programming environment which allows switching between programming tasks and automatically builds up their context. These approaches however are application-specific, and only support aggregating a limited set of document types. TAGtivity [27] allows tagging any resource within a PC environment. While tags don't necessarily need to represent tasks, they were shown to be used in such a way, allowing to maintain multiple working contexts. TaskTracer [9] monitors user's interaction on a computer and associates them with tasks. A user interface provides an overview of the created task context. What many of these tools neglect is that *planning* tasks plays a crucial role in knowledge work as well. The user needs to be able to prepare, structure and reflect on future work. The widespread adoption of electronic calendars and to-do lists indicate their value in task management. However, these tools are intrinsically disconnected from the actual resources needed to start work on them [33].

File Management

Whittaker describes three *information curation processes*: keeping, management and exploitation [35]. People preserve large quantities of emails, bookmarks and personal files and organize them in ways that will promote future retrieval. However, information items are stored, managed and viewed from within application-specific collections, causing *information fragmentation* [6]. Not all information items are purely informative. Some are *action-oriented*, like emails. They require the recipient to do something, possibly before a certain date. They are generally kept around as a reminder. In order to alleviate the overhead of organizing files, time-based organization has been brought forward in previous research. Lifestreams [10] was the first to do so, providing a simple stream of documents over time. TimeScape [28] extends on this by integrating with the desktop interface. Time travel allows going to past and future states of the desktop. Only those documents used or placed at a certain point in time show up. Haystack [1] provides users with the ability to construct individualized information collections by creating relations between separate information items. To simplify file retrieval, Leyline [12] offers information about the creation, use and sharing of documents and their context – *provenance*. Despite support for simplified access to the underlying file hierarchies, in essence files are a remnant of the original desktop metaphor. Users are forced to mentally connect window representations to the files they represent. When restoring window configurations users are still confronted with finding all the related files.

Integrated knowledge work

It is clear that task, window and file management are three practices intrinsically linked to each other. As depicted in Figure 2 supporting the three practices separately causes conflicts once they overlap. Window managers support lightweight *multitasking* but are unaware of the specified tasks in traditional task management tools. *Interruptions* often lead to new tasks or revisitation of old ones [17]. Handling them requires opening up the relevant windows. *Task-centric resources* are often grouped together in the file system. Folder hierarchies sometimes act as kind of project plans [18]. *Action-oriented* items like emails or temporary files are used as reminders [35]. Additionally, email has been shown to be the primary source of new events entered in a calendar, which in its turn is not only used for planning, but also for reporting purposes and recording memorable events [33]. Lastly, information *curation* is handled differently per application, causing *information fragmentation*.

What unifies these different practices is the actual work they set out to support. Studies have shown that people organize their work in higher-level thematically connected units of work, often referred to in literature as *tasks* or *activities* [3, 8, 13]. By constructing tool integration around a computational representation of activities, a lot of *meta-work* can be offloaded to a centralized *activity management* system. To fully support all knowledge worker processes, such a system needs to integrate with the core aspects of task, window and file management.

Activity-centric systems

Prior systems have incorporated *activity management* into the desktop interface. Within these systems, files, applications, communication and collaboration are structured within activities. Activity-centric systems allow the user to switch between different ongoing activities by suspending and resuming them, thus facilitating multitasking. ABC [4], e.g., replaced the Windows Taskbar with an Activity Bar, showing the current list of activities and allowing to easily switch between them. Integrating with Apple OS X, Giornata [34] follows a similar approach, informed by activity theory. Activities can also be annotated with optional freeform tags. Other activity-centric systems such as Activity explorer [11] and co-Activity Manager [16] have explored how communication alerts can be tied to the activities from which they arise, automatically linking notifications to the right context. Some systems have extended the scope of managing activity contexts to some degree including other practices like planning in UMEA [21] or automated archival over time in TimeSpace [23]. However, most of these systems provide little or no support for the entire *activity life cycle*: the source of an activity, transitions and interrelations with other activities.

Going back to the seminal work of Rooms [15], *Laevo* allows users to structure work within *dedicated workspaces*, representing activities. Laevo differs from existing activity-centric systems in that it aims to support the entire life cycle of activities, focusing on all practices which influence them over time. These include short- and long-term planning, archiving, multitasking and interruption handling. As the activity goes through its different states, its context is preserved, and updated accordingly.

THE ACTIVITY LIFE CYCLE

Prior research has explored the full extent of activity context based on theories of cognition and observations of real-world practice. However, by reviewing literature on PIM tools we conclude that it is equally important to investigate how information is passed from one tool to another. Within the context of activities this means investigating how the activity context is created, how it is used, and how it changes over time. An evaluation of Giornata concluded *"the system makes no distinction between dormant activities and those that are completed, which raises further research questions about how individuals think about the stages in an activity's lifecycle"* [34]. In this section we describe *the activity life cycle* (Figure 3), a model representing six fundamental practices that influence the state of activities over time and frame them within the three fundamental processes of knowledge work: *archiving*, *multitasking* and *planning*.

Knowledge work processes

Multitasking is a common process which involves switching between, and managing a large number of windows, files and other resources that are associated with different activity contexts [8, 13, 17]. Within activity-centric computing this is supported by allowing the user to aggregate the related resources into meaningful structures reflecting the ongoing parallel activities, and providing support to easily switch between them. Restoring the entire work context is automated,

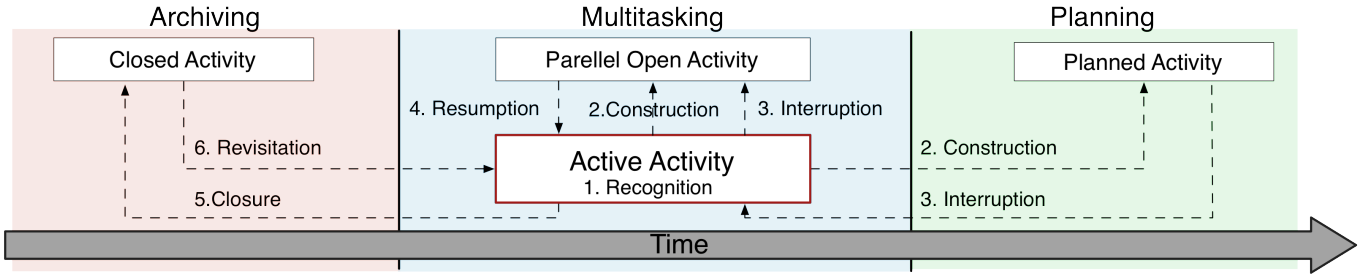

Figure 3. Modern knowledge work consists of archiving, multitasking and planning. Six fundamental practices, related to these processes, determine how an activity evolves over time.

minimizing reconfiguration work the user otherwise has to do manually. There are two important *archiving* practices, filing and piling [25]. Structuring work within activities could be seen as filing, while documents can be piled without too much thought within the activity context. Providing strong provenance cues for activities helps in finding them when their corresponding piles need to be retrieved. Lastly, short- and long-term *planning* are essential parts of knowledge work which shape future activities. When planning activities, a context can already be built up to be exploited once the activity starts.

Activity states and transitions

Activities can be in three different states: *open* when part of the current multitasking session, *closed* when work has been discontinued and *planned* when intending to work on them at a later time. When the user's attention is directed towards an activity it is *active*. At any given time only one activity can be active. There are six practices which cause activities to emerge and change state. (Figure 3)

1. *Recognition*– Prior to constructing activities, users need to form a mental model of them. General activity awareness can reduce occasions where activities become intertwined.

2. *Construction*– There are no clear borders which fully define activities up front. The overhead of constructing, updating and moving resources in between activities should be minimized, supporting ad hoc and post hoc activity creation.

3. *Interruption*– External or self-interruptions can lead to switching to a different activity. Interruptions carry context, which can be incorporated into new or existing activities, supporting *construction*. Handling interruptions effectively can improve *recognition*.

4. *Resumption*– When resuming an activity, it's context should reflect the user's mental model before the activity was interrupted. Quick resumption is essential for multitasking.

5. *Closure*– When an activity is no longer relevant, it should be possible to easily take it out of the ongoing multitasking session without losing it's built-up context.

6. *Revisitation*– Although activities can become irrelevant, they might need to be revisited shortly, or even restored at a later point in time. *Recognizing* activities helps in easier revisitation.

LAEVO

Laevo is a *temporal activity-centric* desktop interface for Windows 7 and 8. It is designed for personal use on a single computer in support of office work. Similar to other activity-centric systems [4, 16, 34], it allows users to structure work (applications and files) within *dedicated workspaces* in order to reflect separate activities. When working within a workspace, Laevo's user interface is kept to a bare minimum; no new interface elements are introduced except for a system tray icon through which the user is notified of incoming *interruptions*. Only when opening the activity overview, the user is presented with a full screen user interface through which activities and interruptions can be accessed and managed.

Dedicated activity workspace

When working on a particular activity, only the relevant activity context is visible, which helps reducing information overload and keeping focus *(recognition)*. This is achieved by setting up a virtual desktop per activity, thus still supporting all the existing tools the user is used to (Figure 4). The context of the activity automatically arises from the user working within the desktop workspace; windows which are opened within the workspace are assigned to the activity it represents *(construction)*. When the user switches to a different workspace, all open windows of that context are restored *(resumption)*. In order to organize files within activities, each activity has access to an *activity context library* (Figure 4A). This is a standard Windows library, similar to "Pictures" or "My Documents", which is always accessible from the side menu in Windows Explorer. The content of the library is contextualized as it only shows files added from within the specific activity. The files are physically located in a folder which is automatically created for each activity *(construction)*, reflecting its date and name, e.g. "9-11-2013 Chi 2014 paper". This supports the common file management practice of structuring files within folders representing tasks [14]. Users can still use their existing folder hierarchies by adding new folder locations to the library. There are no clear-cut borders in between activities: while working on one activity the users might have started work on another. Therefore the system provides a flexible mechanism by which windows can be moved in between different workspaces *(construction)*. Similarly to how files are cut and pasted, shortcut keys can perform these operations on application windows once they have focus. Cut windows disappear immediately and are reassigned to the workspace in which they are pasted. Several cut operations can be performed in a row, placing them on a stack until a paste op-

eration processes them all. Other shortcut keys allow quick access to the activity context library, creating new empty activities, closing the current activity or switching between the last two accessed ones.

Users can always see at a glance whether something requires their attention; Laevo's system tray icon lights up yellow when new *interruptions* arrive (Figure 4B). The visualization is intentionally kept to a minimum in order not to disrupt the flow of the user. Users decide for themselves when they want to address interruptions, which they can do by opening up the activity overview.

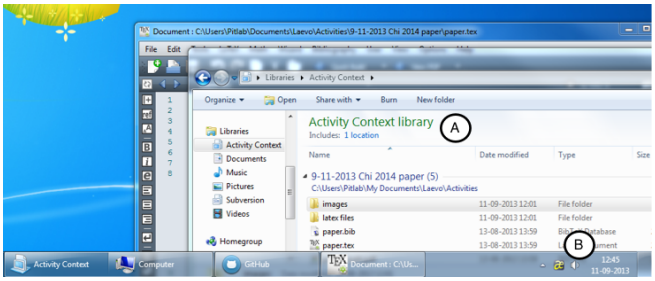

Figure 4. The activity context library (A) and tray icon (B) within a dedicated activity workspace.

Activity overview

We provide an overview of activities in *time* and space, highlighting the *activity life cycle* (Figure 5). At a glance users can see when activities were open or are *planned*, while spatial organization, colors and icons can be used to identify activities or indicate relationships between them. This can be a powerful cue for recall by invoking episodic memory [24], facilitating *revisitation* of old activities. A full screen representation hides the low-level details of the work the user was working on, and is intended to help users reason on a higher abstraction, contrasting the ongoing activity with surrounding ones *(recognition)*. The timeline is the "heart" of the application, and can be accessed at any point in time using a reappropriated keyboard button, CAPS LOCK [1], which allows quick access to other activities *(resumption)*. It also acts as a modifier key to access all other shortcut keys. Alternatively the user can double-click the tray icon to access the overview. The granularity of activities can vary drastically; some lasting weeks, others mere minutes. Therefore, the timeline supports continuous panning and zooming in order to show any arbitrary time interval. Depending on the zoom level, the semantically most meaningful labels are shown (e.g. months or days). A one-dimensional representation of time, as opposed to a more traditional calendar layout, emphasizes the concurrency of activities *(recognition)*. Lastly, applying a perspective transformation has two advantages. It saves screen estate without hampering readability too much, and it strengthens the presentation of time following the common conceptual metaphor of expressing time as a physical path in space. E.g."approaching the end of the year", or "leaving days behind".

Figure 5 highlights the different components on the overview screen, as well as the different *states* activities can be in. An activity can either be *open* to represent ongoing work (Figure 5A), *closed* when finished in the past and now *archived* in time (Figure 5B) or *planned* when a time slot has been assigned for it in the future (Figure 5C). Activities are represented by rectangular areas which stretch along the timeline horizontally, indicating when the working context was open. Only one workspace can be *active* at any given time (highlighted with a yellow border on the timeline), but several can be open simultaneously, allowing for *multitasking*. An activity is made active simply by clicking on it, after which the overview is hidden and its dedicated workspace shows up. Yellow lines along the bottom of a workspace indicate when they were active (Figure 5I), but this visualization can be disabled when deemed too distracting. Users need to change activity states themselves, which increases their awareness about them *(recognition)*. Closed and planned activities can be opened through a hover menu, positioned near the mouse (Figure 5H). Doing so brings the activity into the current multitasking session, visually represented by stretching the rectangle up to a line which indicates the current time. Activating an activity does not open it, so it can be inspected without changing its state. The same menu is used to edit and remove activities, although removing them is discouraged. Instead, the system encourages *archival* over *time* by simply *closing* activities. To reinforce this, an ongoing activity can only be removed after it has been closed, and when it does not contain any open windows. Past activities that do have windows open in them are displayed with an orange border.

When starting Laevo, all windows previously not associated to an activity are assigned to a permanent *home activity* (Figure 5E). It can be used as a workspace for smaller, less important tasks which aren't immediately *recognized* as an activity, or aren't activity-specific.

The user can adjust the name, icon and color of each activity in a pop up menu (Figure 5G) which is opened from the hover menu. As the activity is edited, changes are reflected immediately in the pop up menu as well; it's background color and icon in the top left changes. From the timeline, activities can be dragged up and down, allowing to organize them vertically. This combined with the *temporal* dimension in which they are displayed gives plenty of possible visual cues to identify them, following the redundancy gain principle. To allow for easy *construction* the activity names can be changed directly from the timeline as well; the labels act like ordinary input boxes.

To-do list and interruptions

Although future activities can be *planned* accurately on the timeline, there is also a need for less structured *task management*. Laevo allows the user to create *to-do items* which behave the same way as activities, except that they are always visible on the overview screen (Figure 5F). Only their icons and colors are shown to save space. However, hovering over them shows their name, as well as a menu to edit or remove them (Figure 6A). Simple drag operations allow users to rearrange to-do items to reflect their priority.

Once a user decides to start working on, or plan an activity, the to-do item can be dragged to the timeline. Dropping it

[1] an idea borrowed from humanized.com/enso

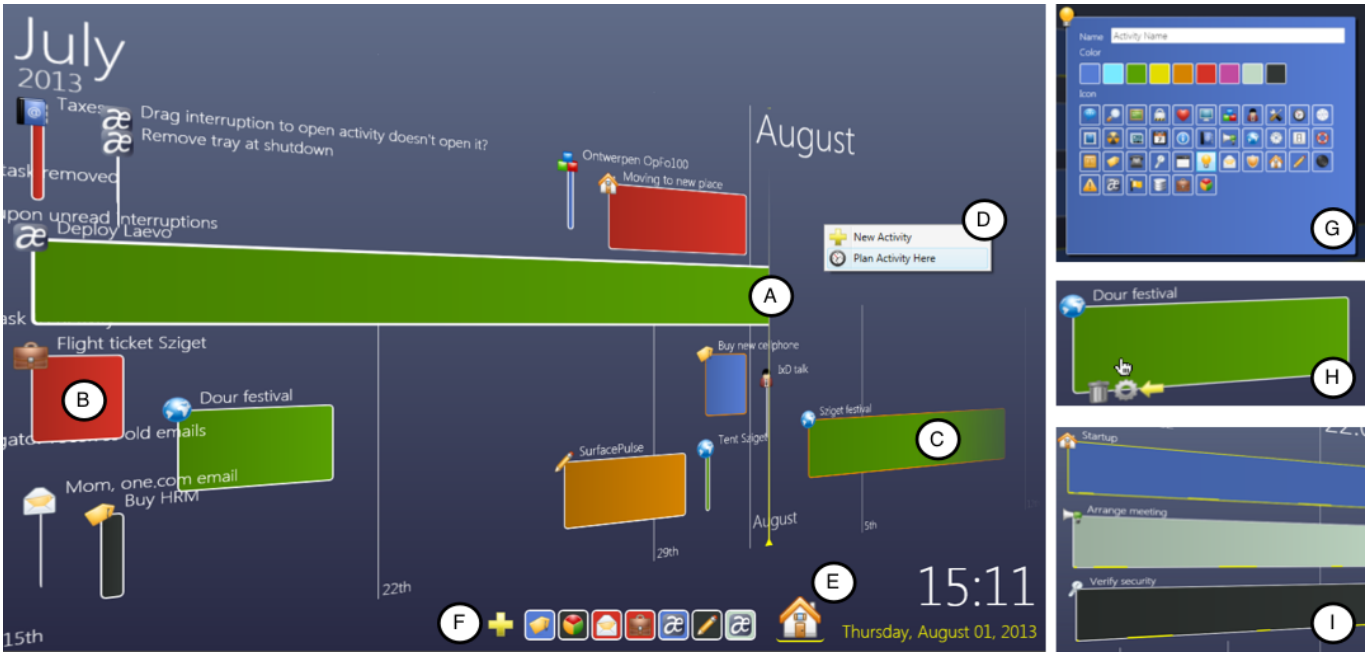

Figure 5. The activity overview through which activities are accessed and managed. It displays (A) open ongoing activities, (B) archived activities, (C) planned activities, (D) a context menu through which new activities can be created, (E) the home activity and (F) to-do list. The three detailed images show (G) the popup menu to edit activities, (H) action buttons when hovering over an activity and (I) attention span lines.

in front of the current time opens it, while dropping it behind plans it at that particular position (Figure 6B). Alternatively to-do items can be dragged to existing activities, merging their contents. This includes their windows, as well as the set up paths of the activity context library.

Laevo features a centralized notification system for *interruptions*, well integrated with the rest of the system. It does so by equating interruptions with to-do items. They are added to the to-do list but can be distinguished by their flashing yellow border (Figure 6A). When within a workspace, the tray icon lights up as long as there are unattended interruptions. Opening them for the first time opens up their context with the appropriate application. E.g. when an email is received it shows up in the to-do list with the subject as name; clicking on it opens up the email message. Using the normal to-do list functionality this allows users to either handle interruptions in place (from within its workspace), merge them with existing activities, open, remove or plan them. This simplifies activity *construction*.

Figure 6. Interruptions, e.g. emails (A), arrive in the to-do list and are highlighted. To-do items can be dragged to the time line to start work on them or (B) plan them.

TECHNICAL IMPLEMENTATION

In order to integrate with Windows 7 and 8, we created a toolkit through which Laevo interfaces. This component-based *Activity-Based Computing (ABC) toolkit* handles common functionalities, expected from an activity-centric computing system. It's comprised of a virtual desktop manager (VDM) which relieves Laevo from having to do custom window management; an interruption handler which aggregates interruptions received by applications; and activity services which expose operations in an application-agnostic way (Figure 7). We envision to keep expanding on this toolkit by adding new components as new requirements arise. It serves as a basis to build new activity-centric systems on. When using the toolkit, no direct interfacing with the operating system is required. Some components provide extension points in order to support applications unaware about the activity-centric system. We are working on an interface with which custom-built applications will interface in order to make them fully *activity-aware*. Both the toolkit and Laevo are written in .NET. Laevo uses Windows Presentation Foundation (WPF) for its front end, and the toolkit uses P/Invoke to communicate with the operating system.

The VDM offers a very minimalistic API. Through a virtual desktop manager an infinite amount of virtual desktops can be created. Besides switching between and merging desktops, the API also supports cut and paste operations on windows. To allow restarting activity-centric applications without losing window configurations, the state of the VDM can be persisted. All open windows previously handled by the VDM are reassigned to the correct desktops upon restarting.

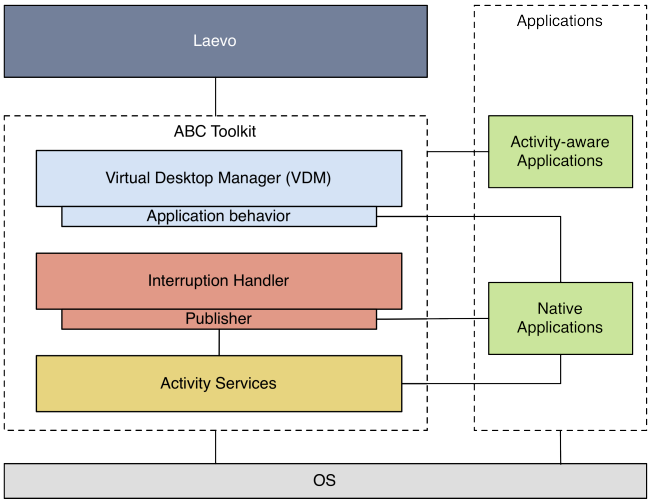

Figure 7. Laevo builds on top of an Activity-Based Computing toolkit which handles the low-level communication with the operating system. Native applications need to hook into it's subcomponents while activity-aware applications connect to a common API.

Through configuration files application-specific behavior can be defined, e.g. which windows should be cut during cut and paste operations. Lastly, a debug client which communicates with the VDM through inter process communication can be used to see details about the managed windows, useful when setting up application-specific behaviors.

The interruption handler allows applications to publish interruptions which will be integrated into the activity-centric system. .NET's Managed Extensibility Framework (MEF) is used to provide an extension point for application plug-ins. Besides publishing interruptions these plugins also define how the related context should be opened once the user decides to look at them, e.g. opening up an email. A plugin was written which regularly checks Gmail's Atom feed for unread emails. Using the activity services from the toolkit, the email is opened in a new tab in the default web browser.

FEASIBILITY STUDY

To explore the *feasibility* of a temporal activity-centric system we conducted a one-day in situ field study with 12 participants using an initial prototype. In this field study, we exposed end users to the core time-based features of Laevo in order to assess its perceived usefulness and usability before moving into a larger scale field study. Additionally initial stability issues could be identified.

Experimental Setup

12 participants (age ranging from 23 to 55 – 3 females) were recruited to participate in the experiment. Participants came from various backgrounds including pharmacy, clerk, sales, software development and social work to represent a broad range of knowledge workers. The only precondition set was that all participants were required to use either Windows 7 or 8, having a minimal knowledge of its operation. The experiment was conducted *in situ*, which means that users could complete the experiment on their own personal computer and in their own work environment.

Method

Laevo was sent to all participants as a simple executable, including a detailed manual on how to set up the system and an explanation of its different functionalities. Users were instructed to use the system during one full working day as part of their day-to-day work, without any restrictions on when to use it or for which tasks. Users that could not use the system for a full day (8 hours), were requested to continue using the system the next day until the amount of 8 hours was reached. Upon completion, users were asked to fill out an all-positive System Usability Scale (SUS) [19] which was used as a starting point for an in-situ interview to elaborate on the results and allow for open-ended comments.

Results

The average SUS rating for Laevo was 83.5 (SD = 8.23, N = 12). Four participants had prior experience with virtual desktops. They were asked to rate their favorite virtual desktop environment, resulting in an average rating of 76 (SD = 9.52, N = 4). One user used mDesktop and four others Ubuntu's multiple desktops. One user explained why she considered Laevo's overview screen to be clearer and more useful:

> *"I regarded [Ubuntu's VDM] as several screens between which you can switch, [in Laevo] I saw it more as a distribution of my activities." – Ubuntu user*

All users except one reported having to work on four or more activities simultaneously on a regular basis. This corresponds to the scenario we set out to support. Some users mentioned the interface would be even more useful over longer periods of time when the need arises to restart work on an old activity. Participants generally liked the user interface (as indicated by the SUS rating) and no significant usability problems were identified. There were however a number of additional functional requirements that emerged from the feedback of the users. First, users requested the ability to plan activities on the timeline, and to extract working hours from them. Secondly, many users had one permanent activity open they used for ad hoc work since the overhead of creating an activity was larger than executing the actual task. Therefore a default home activity was added to the design which became the default starting point when running Laevo. Next, to allow for more granular activity management, functionally was added to more easily view, open and close activities. Finally, expert users requested keyboard shortcuts to easily switch between the two last actively used activities.

FIELD STUDY

After we verified the *usability* and overall stability of the system in a prior study, we introduced a number of additional features derived from the activity life cycle. This second prototype was deployed during a two-week field study. The goal of the study was to assess the middle- and long-term effects of using Laevo.

Experimental setup

6 participants (age ranging from 28 to 55, – 1 female) were recruited to participate in the experiment. Again participants were recruited from a broad range of backgrounds, including consulting and software development, to represent different types of knowledge work. As in the prior experiment, we required participants to have experience with either Windows 7 or 8. Again, the experiment was conducted in situ, meaning that the system was deployed on their personal computer, used in their own work environment. Only two of the 12 users from the feasibility study participated in this second study. The other 4 participants used Laevo for the first time.

Method

Users were sent a link to a blog post containing an installer and a complete manual of Laevo. Participants were requested to use the system over a period of 14 days during their day-to-day work. During this period, they were asked to keep a diary, in which they had to add an entry once each day. The entry was based on a number of predefined questions on how and if they used the system that day, with a particular focus on any special events. Additionally, usage data from the system was automatically collected throughout the experiment. At the end of the experiment participants were requested to anonymize any sensitive information before submitting their data and diary. From this data, a local representation of their timeline could be reconstructed for analysis. Finally, after the experiment was completed, participants were invited for a semi-structural interview in which their experiences with the system were discussed. Initial questions were individually outlined beforehand based on their diaries.

Results

All participants experienced great benefits by structuring their work within the context of activities, confirming prior findings of in situ field studies which have evaluated activity-centric systems. Once activities are set up, people like having only those things for the task at hand visible as it helps them in keeping *focus*. Participants experienced losing less time when switching between parallel tasks since the right folders and files were still open. Without Laevo, extreme *filers* (P1, P2, P4) ordinarily closed windows in order to coordinate many parallel tasks.

> *"I'm getting used to not having a cluttered desktop. I made an activity for something that only took about one hour, but this allowed me to focus on the task at hand." – P1*

When asked to elaborate on where this focus came from it became clear that Laevo made participants more aware about their activities. The activity overview played an important role in this. Participants argued that each time they switched between activities they were exposed 'at a glance' to all ongoing and planned work, which P5 stated as being an advantage over a to-do list on paper which can easily be ignored. This allowed them to make more conscious decisions about which activities to prioritize. As mentioned by one user:

> *"By explicitly defining your activity, you are [...] forced to reflect on what is your current active 'task', [which] seems to make you less distracted. When you do switch to something else, it becomes a very conscious choice." – P6*

Analyzing the participants' timelines indicated that for all users except one (P3) the majority of activities were concrete instantiations aimed towards a certain goal, as opposed to 'types' of activities like 'programming'. Noteworthy, two users (P2, P3) who were using Outlook instead of Gmail, and thus weren't receiving email interruptions, created long-running 'check email' activities.

> *"Using other [VDMs] I structured my work according to [type of work]. What distinguishes Laevo from other [VDMs] is it is inviting to organize [work] as concrete activities." – P6*

The activity model provided by Laevo was appropriated by users to support different activity *scopes*, ranging from only having short hour-based tasks (P2, P6), to long term projects that lasted several weeks (P3, P4), to a combination of both (P1, P5) (Figure 8).

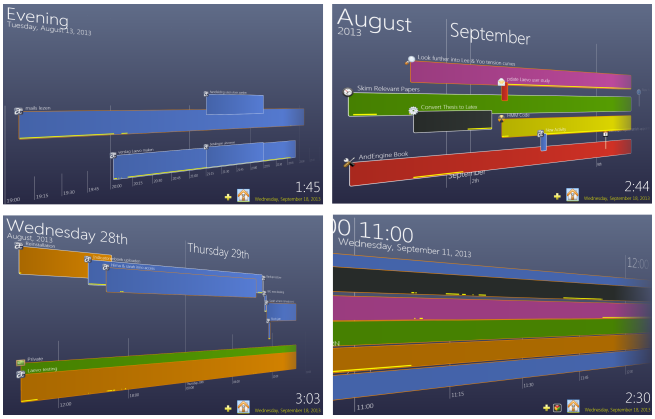

Figure 8. Different activity scopes.

One user (P5) created a number of short tasks at the start of each day, meticulously planning which work needed to be done. As the day progressed he used the activity timeline to check whether he was still on schedule, reprioritizing activities where needed. Another user (P4), however, argued that activities only made sense to him for long term projects. As part of his work, he created activities for each of the clients he was working with. For all smaller tasks he used the home activity, or simply handled them inside the currently open one. In general, all participants saw the home activity as a catch all environment, often used for quick and dirty work or private activities, thus keeping actual work separated from 'daily clutter'.

All participants that eventually created short-term activities mentioned a *'learning curve'* they had to go through when using Laevo. After a while, P1 and P2 adopted a 'nothing ventured, nothing gained' attitude towards creating activities. The up-front configuration work initially seems like an overhead, but over time users start to see several advantages when doing so; *less clutter*, increased *focus*, *productivity* and *efficiency*.

"[...] the more separate activities I start in Laevo, the less I'm tempted to quickly do another activity within an existing one, the more fluently and efficient I can work. You have to 'learn to use' Laevo – as is the case with everything." – P2

Participants incorporated the system to varying degrees into their existing work practices, depending on how much overlap there was with the other tools they used. Before using Laevo, P2 used to write to-do items on a piece of paper. She first experimented with planning activities in the future in order to remind her of them, but afterwards started using the built-in to-do list instead. P1 opened up entire projects within to-do items as placeholders for side-projects he wanted to start working on when he had some spare time. Another participant (P5) preferred Google's task list since he could easily access it from his mobile phone. However, at the start of each day he manually transferred the to-do items with the highest priority to Laevo's timeline, planning his day. He preferred Laevo's overview over Google's task list. Lastly, P6 did not use the to-list nor the incoming email interruptions since he kept an elaborate to-list on paper, which he divided in different zones by priority.

Similarly, participants used different approaches to construct activities. P5 had set up emails to arrive in his to-do list. A request to update his daily report of Laevo lead to opening it up as a new activity, while a reminder email by Google was dragged to the corresponding time on the timeline. Another participant (P4) who had not set up email interruptions manually cut and pasted email windows to the corresponding activities, effectively using them as reminders within his long-running projects. P2 manually introduced to-do items, which lead to the creation of activities the next day. Analysing the timelines of all participants (e.g. Figure 8), it was clear that users applied a wide variety of color schemes and physical layouts.

Laevo currently provides limited support for the revisitation of activities. Previously closed activities that are reopened are shown on the timeline as a continuous visualization which most participants found to be an incorrect representation. The original design rational behind this feature was to create a connection between the two points in time the activity was used. The yellow lines indicate when the activity was active. The users, however, argued that it should not be represented as a continuum but rather as *multiple instances* of that particular activity.

Three out of six participants (P2, P5, P6) continued using the system after the two week study. Two other users mentioned they would continue using Laevo in case stability issues would be resolved (P1 and P4) and in case integration with Outlook would be provided (P4).

DISCUSSION

In this paper we presented a review of different knowledge worker processes as reported in prior research, covering many different domains. We reviewed different practices covered in PIM literature, as well as different approaches to tool integration, with a particular focus on activity-centric computing. From this we derived a model for the *activity life cycle*, which lead to the creation of a novel *temporal activity-centric* desktop interface, Laevo. Two in field studies of the system provided interesting results and allowed us to describe a number of lessons learned and opportunities for future work.

Similarly to other activity-centric systems, the dedicated workspaces provided by Laevo were appropriated by users in different ways, ranging from *long-term* projects to *ad hoc* tasks. Additionally, Laevo brought forward to-do items and planned activities as an important mechanism to structure future work. The explicit full screen time-based overview made users more *aware* about their ongoing and planned activities, empowering them to make *conscious* decisions on how to structure and organize their work. Participants reported being more *focused*, *productive* and *efficient*.

Laevo envisions a full integration of *task, window and file management*, and within this paper we have demonstrated and evaluated several interaction techniques working towards this goal. For example, the cut and paste window operations allowed users to more easily create and reconfigure activities, without having to move underlying resources around. It brings window management closer to file management, raising interesting questions for further research how window management can be re-envisioned to outgrow its original purpose. One of the core visions behind Laevo is to increasingly move away from files, as their main intent of persisting information could be replaced by persisting *window configurations*, represented as activities. Structuring work within the context of activities, and providing strong provenance cues of their life cycle and interrelations with other activities, is how we envision to support what Whittaker refers to as *the curation lifecycle* [35].

ACKNOWLEDGEMENTS

Anonymized

REFERENCES

1. Adar, E., Karger, D., and Stein, L. A. Haystack: Per-user information environments. In *Proceedings of the eighth international conference on Information and knowledge management*, ACM (1999), 413–422.

2. Agarawala, A., and Balakrishnan, R. Keepin'it real: pushing the desktop metaphor with physics, piles and the pen. In *Proc. of CHI 2006*, 1283–1292.

3. Bannon, L., Cypher, A., Greenspan, S., and Monty, M. Evaluation and analysis of users' activity organization. In *Proc. SIGCHI '83*, 54–57.

4. Bardram, J., Bunde-Pedersen, J., and Soegaard, M. Support for activity-based computing in a personal computing operating system. In *Proc. CHI '06*, 211–220.

5. Bellotti, V., Ducheneaut, N., Howard, M., and Smith, I. Taking email to task: the design and evaluation of a task management centered email tool. In *Proc. CHI '03*, 345–352.

6. Bergman, O., Beyth-Marom, R., and Nachmias, R. The project fragmentation problem in personal information management. In *Proc. CHI '06*, 271–274.
7. Boardman, R., and Sasse, M. A. "stuff goes into the computer and doesn't come out": a cross-tool study of personal information management. In *Proc. CHI '04*, 583–590.
8. Czerwinski, M., Horvitz, E., and Wilhite, S. A diary study of task switching and interruptions. In *Proc. SIGCHI '04*, 175–182.
9. Dragunov, A., Dietterich, T., Johnsrude, K., McLaughlin, M., Li, L., and Herlocker, J. Tasktracer: a desktop environment to support multi-tasking knowledge workers. In *Proc. IUI 2005*, 75–82.
10. Freeman, E., and Gelernter, D. Lifestreams: a storage model for personal data. *SIGMOD Rec. 25*, 1 (Mar. 1996), 80–86.
11. Geyer, W., Muller, M. J., Moore, M. T., Wilcox, E., Cheng, L.-T., Brownholtz, B., Hill, C., and Millen, D. R. Activity explorer: activity-centric collaboration from research to product. *IBM Systems Journal 45*, 4 (2006), 713–738.
12. Ghorashi, S., and Jensen, C. Leyline: provenance-based search using a graphical sketchpad. In *Proc. HCIR '12*, 2.
13. González, V., and Mark, G. Constant, constant, multi-tasking craziness: managing multiple working spheres. In *Proc. CHI '04*, 113–120.
14. Henderson, S. Genre, task, topic and time: facets of personal digital document management. In *Proc. CHINZ '05*, 75–82.
15. Henderson Jr, D., and Card, S. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics (TOG) 5*, 3 (1986), 211–243.
16. Houben, S., Bardram, J., Vermeulen, J., Luyten, K., and Coninx, K. Activity-centric support for ad hoc knowledge work - a case study of co-activity manager. In *Proc. of CHI '13*, 2263–2272.
17. Jin, J., and Dabbish, L. A. Self-interruption on the computer: a typology of discretionary task interleaving. In *Proc. SIGCHI '09*, 1799–1808.
18. Jones, W., Phuwanartnurak, A. J., Gill, R., and Bruce, H. Don't take my folders away!: organizing personal information to get ghings done. In *CHI '05 Extended Abstracts*, 1505–1508.
19. Jordan, P. W. *Usability evaluation in industry*. CRC Press, 1996.
20. Kandogan, E., and Shneiderman, B. Elastic windows: evaluation of multi-window operations.
21. Kaptelinin, V. Umea: translating interaction histories into project contexts. In *Proc. CHI '03*, 353–360.
22. Kersten, M., and Murphy, G. Using task context to improve programmer productivity. In *Proc. SIGSOFT '06*, 1–11.
23. Krishnan, A., and Jones, S. Timespace: activity-based temporal visualisation of personal information spaces. *Personal and Ubiquitous Computing 9*, 1 (2005), 46–65.
24. Lamming, M., and Flynn, M. Forget-me-not: Intimate computing in support of human memory. In *Proc. FRIEND21, 1994 Int. Symp. on Next Generation Human Interface*, Citeseer (1994), 4.
25. Malone, T. W. How do people organize their desks?: Implications for the design of office information systems. *ACM Transactions on Information Systems (TOIS) 1*, 1 (1983), 99–112.
26. Muller, M. J., Geyer, W., Brownholtz, B., Wilcox, E., and Millen, D. R. One-hundred days in an activity-centric collaboration environment based on shared objects. In *Proc. of CHI '04*, 375–382.
27. Oleksik, G., Wilson, M., Tashman, C., Mendes Rodrigues, E., Kazai, G., Smyth, G., Milic-Frayling, N., and Jones, R. Lightweight tagging expands information and activity management practices. In *Proc. CHI '09*, 279–288.
28. Rekimoto, J. Timescape: a time machine for the desktop environment. In *In extended abstracts CHI 1999*, 180–181.
29. Robertson, G., Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D., Meyers, B., Robbins, D., and Smith, G. Scalable fabric: flexible task management. In *Proc. AVI '04*, vol. 25, 85–89.
30. Robertson, G., Van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D., and Gorokhovsky, V. The task gallery: a 3d window manager. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2000), 494–501.
31. Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., Robbins, D., and Andrews, D. Groupbar: The taskbar evolved. In *Proc. OZCHI '03*, 10.
32. Tashman, C. Windowscape: a task oriented window manager. In *Proc. UIST '06*, 77–80.
33. Tungare, M., Perez-Quinones, M., and Sams, A. An exploratory study of calendar use. *arXiv preprint arXiv:0809.3447* (2008).
34. Voida, S., and Mynatt, E. D. It feels better than filing: everyday work experiences in an activity-based computing system. In *Proc. of CHI '09*, 259–268.
35. Whittaker, S. Personal information management: from information consumption to curation. In *Annual review of information science and technology (ARIST)*, B. Cronin, Ed., vol. 45. 2011, 3–62.