Presenting the Past: A Framework for Facilitating the Externalization and Articulation of User Activities in Desktop Environment

Kimmo Wideroos and Samuli Pekkola
Department of Computer Science and Information Systems, University of Jyväskylä,
PO Box 35 (Agora), 40351 Jyväskylä, Finland
{wikikr, samuli}@cc.jyu.fi

Abstract

Work processes are conducted in various contexts and they involve different tasks, interruptions, activities and actions. In all of these, tacit knowledge plays a part. Some part of that tacit knowledge can be externalized and articulated by continuously monitoring the user's activities. Because the desktop environment is an integral part of almost any office work context, we chart the demands the unstructured and discontinuous nature of work puts on the management of desktop working context. We discuss possibilities to augment the user's awareness of his/her desktop working environment by providing a context-aware application that can act as a map-like resource for the user's past activities on the desktop. We propose using temporal information to couple personal experiences with representational, more objective aspects of the context in order to make it possible for the user to express and retrieve subjectively significant activities with a minimal effort. We present an abstract model for designing an application for this purpose.

1. Introduction

Often knowledge management is approached from organizational perspective. However, knowledge management endeavors have faced problems and challenges of changing organizational culture and people's work habits. These can be partly explained through McDermott's [24] characteristics of knowing and knowledge. One of the main characteristic differentiating knowledge from information is that knowing is a human act. Knowing and knowledge, thus, always involves a person who knows. Consequently, because all human acts take place in some context, the issues of context and individuals are of major importance.

Several attempts to conceptualize the context have been launched, especially in context-aware computing. Nevertheless, the emphasis has been on the objective features of a certain type of environment while users' subjective experiences have been left intact. For example, both Chalmers [3] and Grudin [14] criticize context-aware computing on overemphasizing the role of location-awareness at the expense of other kind of information. In Grudin's [14] words:

> *"Temporal context is lost – applications do not generally record or make much use of a person's history. Attentional context is missed – applications do not know where user's attention is focused, whether he or she is busy or open to interruption, talking to someone or on the phone, and so forth."* [ibid. p. 283]

Furthermore, Chalmers [3] points out that the user's experience and history as a part of the user's current context is often ignored in context-aware computing research. The historical aspect of context is important from the personal knowledge management point of view, because "knowledge is what we retain as a result of thinking through a problem, what we remember from the route of thinking we took through the field." [24, p. 106]

In this paper we discuss possibilities to combine personal experiences and certain aspects of the objective, representational context by using temporal information as a reference. We identify a set of issues that need to be considered when designing systems for managing temporal dependencies between one's personal work activities. Because of the scope, discussion is limited to desktop environments. Although desktops are used for multiple purposes and concurrent activities, we argue that they have a poor support for the management of work due to the decoupling of user activities and their contents. This contradicts with Chalmers' [3] suggestion of a general system design goal that interconnects objective representations of system structure with other more subjective and historical representations.

Peirce (as cited in [34, p. 294]) has classified context as actual, modal and intentional. An actual context is a description of a part of the world (at a specific moment). A modal context refers to some possibility relative to what is actual. An intentional context describes what an agent intends in relation to what is actual. Here, the actual context is the part of the user's personal information

0-7695-2507-5/06/$20.00 (C) 2006 IEEE

space that is accessed through the desktop environment, over which the user is supposed to have full control. Although the desktop environment is an important part of the working environment, it is only one part of it [2; 36]. Hence, as the intentional context belongs exclusively to a human agent, there is a fundamental asymmetry between the human and the computer and in their abilities to control the context. In this respect, our approach is strongly human complementary, i.e., endeavors to make computers to collaborate with humans by exploiting the unique abilities of the computer to complement humans [37].

Consequently, we aim at making computers to support the management of temporal dependencies between personal activities. We provide a design for an application which augments the user's awareness of his/her working context by allowing the expression and retrieval of landmarks of actions with minimal effort from the user's side. Landmarks refer to the 'stages' that have substantial influence on the user's intentional context in the course of actions. They include, for example: accomplishing a task, being interrupted unexpectedly, shifting between tasks, recognizing something that might well be valuable with respect to another situation when doing something else, and so forth.

In addition to conscious activities, selective use of automatically recorded usage data can provide a useful contextual background for one's landmarks. We believe that the time-based combination of landmarks and concurrent usage data could turn out to be important entries to the user's personal information space and, further, to the user's history of activities. According to Simon [33, p. 55], users can be seen as designers of their own work: the users can be considered as experts when reflecting upon their activities in the course of time. They recognize the essential landmarks of their activities. Thus, providing users with a map-like resource (i.e. interactive visualizations) to their past history of activities with an emphasis on significant landmarks of actions can facilitate the management of manifold dependencies of individual's work in desktop environment. This also helps the individual to articulate and externalize these activities and tacit knowledge into the organizational context.

2. Need for augmenting user awareness of the context in desktop environment

The following fictitious use case illustrates some of the problems related to the working context in desktop environments.

"John Smith is a software engineer working in a software company. He is responsible for the maintenance of two different software components, A and B.

In the afternoon of an ordinary working day, having just checked out the latest version of the source code of the component A, John is focusing on its specific class implementation. After a while, he is interrupted by a signal indicating that he has new email. He shifts to the email application and recognizes that it is from an important customer. It is an error report concerning the component B which should be fixed immediately. The customer had reported the same error a couple of weeks earlier, and John distantly remembers having fixed something in the code of the component B immediately afterwards. John now checks the date of the earlier email (March 22th), and then shifts to console window to check out the differences between the versions committed in March 21st and 22nd. Unfortunately, there seems to be quite a lot of changes. Now he recalls that after fixing the error, he became 'carried away' and had done quite a lot of other changes and minor fixes in the code before committing the changes to the versioning system. After browsing the code for half an hour, John is finally able to locate the piece of code where he made the error fix. But then the phone rings. It is John's boss asking him to join a meeting he is having with another customer. John heads for the meeting. There he gets a task to send some documents to the customer. On the way back to his office, John receives a SMS from his wife. It appears that their daughter is ill and he should go and pick her up from school as soon as possible.

Next morning John tries to recall what was going on the day before and what he should start with. Ah! Sending the documents to the customer he and the boss met yesterday. Afterwards, John starts thinking about his daughter, about her birthday that is coming soon. Maybe he should buy her a book. John starts the browser, searches for a book, and ends up bookmarking one for his wife as well. Then, John shifts his focus on the component A and continues editing its code."

The described use case is fictitious but it, nevertheless, condenses some real issues, some essential and some more general, of everyday working practices into a short passage.

- Although working environment is always unique, desktop environment is the centre of personal information space and the crucial part of personal knowledge management in most of the cases.
- The contemporary state of the user's personal information space is an end-product of his past interactions and can be seen as an objectified part of the context. As Krishnan and Jones [23] put it, the personal information space evolves over time giving the context for activities the user performs on his desktop. Dourish [6] refers to such an objectified account of context as the representational context in contrast to the interactional notion of context.

- The history of activities and the flow of changes made in personal information space is itself an important part of the subjective and interactional context. Dourish [6] characterizes interactional context as dynamically defined, relational and occasioned property between objects or activities that arises from the activity.
- Although the desktop environment itself is potentially multipurpose and can include specific applications, most of the desktop environments share a set of office applications. The use of the 'conventional' applications is intertwined with the use of more specific applications.
- Interruptions tend to result in discontinuities in the tasks beyond the duration of the interruption itself [25]. In the desktop environment this phenomenon is common, because usually there is a lack of contextual reminders of the recent activities. For instance in the earlier example, John fails to remember the customer's request for the component B, but continues his work on the component A.

These issues frame the information intensive work on a desktop environment. However, they provide only a framework – no attention is paid on the context specific issues; for instance, on the details of the work of the software engineer in the use case. To tackle with these issues, a number of ethnomethodologically driven studies have been carried out [15; 21; 31]. Of these studies unfortunately only a few are on context-aware context.

However, we can learn from existing ethnomethodological studies by generalizing the appropriate finding to the context-aware context. These might include problems or challenges faced by people and ways they act in computerized environments. Next, we will demonstrate this with the help of a study by Ehrlich and Cash [8]. Although they focus on specific professionals, 'search experts', some general issues of information intensive work are identified. Customer support personnel and corporate librarians are familiar with a range of information sources: different types of computerized (i.e. web-based on-line databases, CD-ROMs) and printed materials - and the telephone [8]. However, the tasks the search specialists do vary a lot, for instance in terms of search methods and media. As Ehrlich and Cash pointed out: "other search specialists may range far and wide to pick up tidbits of information which will eventually be put to good use." [ibid. p. 153]. Obviously, when a search task is less structured, when it spans over a long period of time, and when it is occasionally interrupted and includes more work on the desktop, there is a substantially higher risk for the search specialist to lose the working context. In the customer support setting, one of the problems was that an analyst solving a problem and looking up the tracking database for similar cases did not necessarily recognize the match between the documents, thus missing the relevant one [8].

The issue with every day computing is very similar to the ones that stem from the use case of John Smith and from Ehrlich and Cash's [8] studies. In fact, the way Ehrlich and Cash characterize work agrees with the idea of everyday computing. "Work, we have come to believe, is more than tasks and transactions. Making sense of information is a real-time process, and is often both collaborative and emergent." [ibid. p. 164] This reasserts the validity of applying the principles of everyday computing also into conventional working environments with desktops.

Everyday computing emphasises informal and unstructured activities thus making it relevant also to traditional working environments with desktops. The following list includes issues that everyday computing [1] should address.

- Activities may span over a long period of time and do not have a clear beginning or end. For example, a specific task that may seem to have been completed some time ago may require further elaboration.
- Working process is not continuous but might get interrupted, intentionally or accidentally, by the user or by other people.
- Various concurrent activities take place. Shifts between activities occur both in the desktop environment (shifting from one application to another) and in the physical environment.
- Time is an important factor. Besides considering time as an important discriminator, we suggest that temporal information can also bind concurrent events or objects together.
- There is a need for associative information models, because hierarchical information models do not meet the needs of everyday activities*: the models of information for activities* are fundamentally associative. Recent activities may prove out to be a useful source for associations which may be used for directing present activities. Thus, incorporating temporal information, i.e. the history of events and changes made in the personal information space, would be of major importance.

3. Interoperationality

Despite of the fact that systems are not fully interoperable, we are able to operate these systems. In contrast to interoperability, we use the term *interoperationality* to emphasize the role of meaningful human activity in making non-interoperable systems to work seemingly together for some meaningful purpose. This is historical in nature: the meaning emerges in the process of activities. Consequently, for the present purpose, interoperationality

also emphasizes subjectivity. That is, personal working history is meaningful primarily for the person concerned. A chaotic-looking real-world working environment can serve as an illustrative analogy: one's working environment can appear as a total chaos for a colleague, while, at the same time, it can remind its user of some important tasks that should be carried out.

Fischer [10] has argued that the fundamental problem in system design is how to write software for millions of users (at design time) while making it work as if it were designed for each individual user (who is known only at use time). Inspired by the elaboration of the use case above, we argue that designing an ideal solution even for a single person would not be any easier. The difficulty stems from the fragmentation of the (computerized) working environment. Looking at the information systems either from the individual user's perspective or from the perspective of an entire organization, the user-environments or systems are usually made up of several components or pieces of software. Heterogeneity yielding to poor interoperability among systems is a well-addressed problem (c.f. [26]).

Although a new application is designed with its intra-operational use-context in mind, the problem of interoperational use-context remains: the designed system will be only a part of the working environment. For example, although specialists and professionals use their dedicated applications, they also utilize some common programs (web browser, e-mail). We suggest that as far as desktop environment is concerned, augmenting the user's awareness of the working context should address the problem of interoperationality between different applications (c.f.[27]).

Unfortunately, the existing desktop applications restrict the user's awareness of the working context as a whole. To overcome this problem and to address the challenges and demands elaborated in the previous chapter, we chart different alternatives to support and augment the user's awareness across the boundaries of existing systems. In order to rise to the challenge, an application-specific focus is not adequate. The idea of interoperationality underlines several issues that are identified, for instance, in everyday computing approach. In order to concretize and operationalize interoperationality in the desktop environment, relevant technical means are discussed.

4. Desktop environments as platforms for interoperationality

Desktop environments with a graphical user interface support some very general level contextual mechanisms. Clipboard, for example, can be seen as a context mechanism [39]. It has both an intraoperational and interoperable mechanism: contents can be interchanged between applications and within a single application. Model-View-Controller (MVC) [29] architecture separates an application's data model, user interface, and control logic into three distinct components: Model, View and Controller. In spite the fact that single applications seldom implement the MVC model in its purest form (Smalltalk, c.f. [12] applications making exceptions), the MVC model frames the desktop operation system itself as a whole. The file system corresponds to the Model, the desktop to the View and the operating system to the Controller. The file system can be deemed as the storage for persistent, long-term representations of different applications' data models and the desktop as the 'container' for applications' views. The operating system acts as the controller over the whole system (e.g., dispatches the user-events and system-events).

Window-based user interfaces generate user interface events concurrently with their normal operations. From the application usage and usability points of view, this information can be regarded as a potentially fruitful source of information [18]. Further, as the stream of user interface events provides a detailed usage history, it is also a useful source for applications that support the user's context-awareness. Thanks to the role of the operation system as a centralized controller of event dispatching, these events can be captured without a need for modifications to the existing applications.

The fact that user interface events (UI) are rich in detail can cause problems, as well. A great many of the UI events are application specific providing scarcely any semantic meaning without appropriate context. For example, 'MOUSE_PRESSED' may occur when the mouse cursor is on a button labelled 'Search', or when a scroll bar is moved, or when one clicks the background. Nevertheless, there are UI events that inherently include some contextual information, namely the events indicating focus shifts between applications (referred to as '*focus_changed*' events from here on). By confining to such events that explicitly signal user-initiated shifts among applications (i.e. interoperational events), the voluminous stream of events can be simplified.

In addition to user interface events, the file system events could be considered as beneficial for the pursuit of interoperationality. Namely, binding the currently focused window (i.e. the window that has been brought in front of the other(s)) together with events from the file system are subsumed by a more abstract event level. For example, the sequence of user-interface and file system events '*focus_changed*: MS WORD', '*accessing_file*: CV.DOC' and '*changing_file*: CV.DOC' during a short period of time constitutes a semantically richer event, namely '*editing_document*: CV.DOC'. The selective combination of the micro-level user interface with concurrent events from the file system as discussed earlier, on one hand, reduces

the amount of data to be captured and, on the other hand, yields contextually richer events. Consequently, we suggest that framing the desktop environment through the MVC model constitutes a promising operational base for implementing interoperational context-aware applications.

5. A Model for a context-aware application

A model for an application that supports awareness between different applications is presented in Figure 1. The emphasis is on interoperationality: capturing and representing the past activities of a user. In this, we take advantage of the operational means that the desktop system provides and were discussed earlier.

To articulate the idea of the application, we have used components from a context-aware application toolkit from Dey et al. [4] (they bind the framework with the physical context and emphasize sensor-data, but we suggest that the toolkit is applicable also as a framework for conventional desktop applications augmenting the user's context-awareness): *Context widgets* support a uniform interface that allows the applications to acquire contextual information; *Interpreters* generalize contextual information by raising its level of abstraction; *Aggregators* gather logically related information relevant for an application and make it available within a single software component.

The *context-aware application* model decouples the collation of usage data and presenting the data to two separate components or submodels: the *context-aggregator* and the *context-view*. These components are discussed in more detail below.

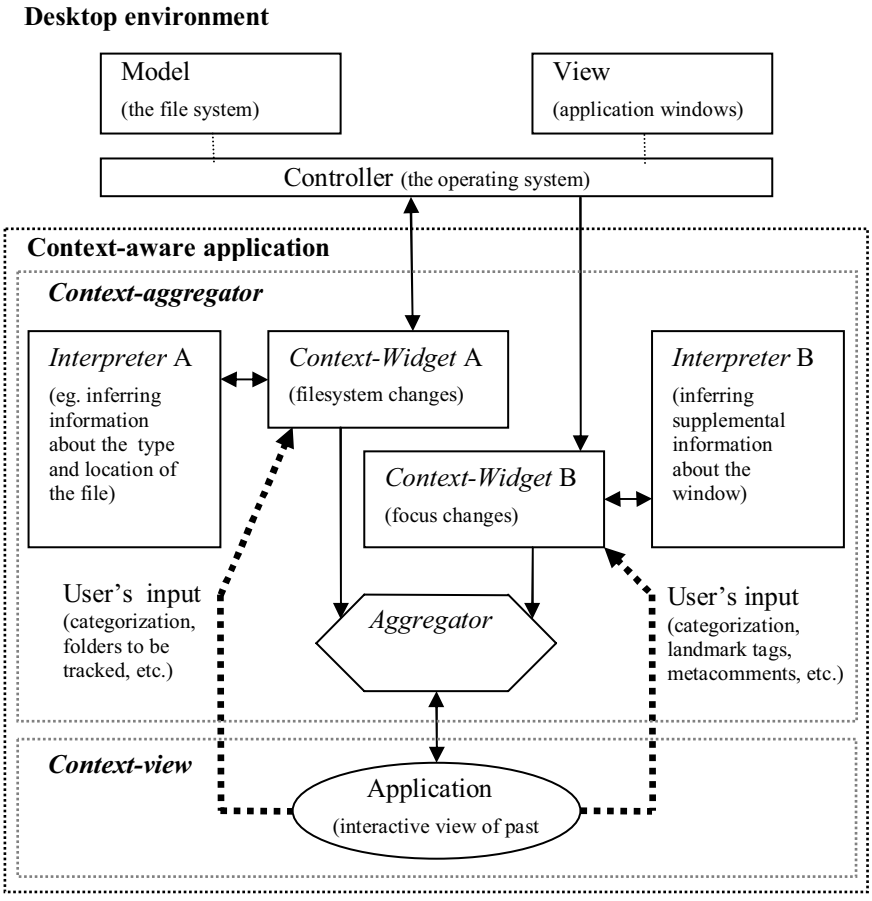

Figure 1: The model of the application.

5.1. Gathering the past activities: context-aggregator

The context-aggregator encapsulates interfacing with the operating system and acts as a repository for the past activities of the user.

The *Context-widget A* gathers the file system events and the *Interpreter A* conducts file specific abstractions on these events (e.g. if the files concerned are internet temporary files, it infers the URL and/or the topic of the page). The *Context-widget B* keeps an eye on focus changes in the user interface and uses the *Interpreter B* to attach extra information about the context (e.g. infers the name of the focused window). The extra information attached depends on the type of the activity: a landmark of actions is supplemented with more detailed information than the other activities.

On the general level, the way the *Aggregator* operates is similar to the techniques for synchronization and searching applied in usability research. These techniques are based on the idea of synchronizing and cross-indexing UI events with other sources of data [18]. In the case of the *Aggregator*, the focus-information from the context-widget B is bound with the corresponding file changes events from the Context-widget A according to their temporal relation.

In the following a sample user-scenario is provided in order to illustrate the functionality of the *Aggregator* on an abstract level. Let *WF(t)* denote the window that has gained the focus at the *t* and let *FS(t)* be the file system event with respect to time *t* (see the corresponding timelines in the Figure 2).

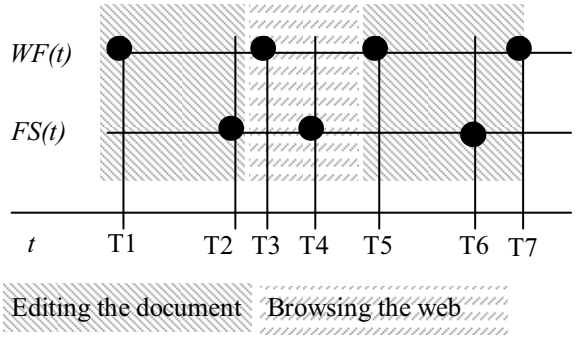

Figure 2: Binding focus and file events together.

Now, consider the following scenario: the user launches an application, let's say a word processor, at the time *T1*. Consequently, the application window gets the focus and the corresponding event *WF(T1)* is recorded. After writing for a while, the user decides to save the document: the file creation event is recorded in *FS(T2)*. Further, at the time *T3* the user launches a web browser for checking an article. This causes a new focus changing event at *WF(T3)*. After the user has entered the URL, a web page is fetched – and the corresponding file appears in the temporary internet files folder. This change in file system is recorded as *FS(T4)*. At the time *T5,* the user shifts back to edit the document, and the word processor application gains the focus – *WF(T5)*. Finally, the user saves the document and quits the word processor: events *FS(T6)* and *WT(T7)* are recorded respectively. The gray blocks in Figure 2 illustrate how the *Aggregator* binds the focus events with the corresponding file events.

The interaction between the user and the context-aggregator happens through the context-view. The user's input is threefold: preferences, categorizations, and landmarks. Preferences include, for instance, the folders to be tracked. Categorizations are states (e.g. should the context-aggregator be recording the user's activities or not) or identifiers (e.g. 'Work', 'Leisure', 'Critical computing article') of long-term activities. Categorization information is of use for both of the context-widgets. The user's request for a landmark 'forces' the *Context-widget B* to make a 'focus changed' record and supply it with supplemental information about the context.

The context-aggregator model (Figure 1) does not describe the contents of the contextual information attached to a landmark of actions. Instead, it encapsulates the logic of selecting relevant contextual information in the *Interpreter B*. In the following, a general and rather straightforward approach to the implementation of the *Interpreter B* is described. In this approach, each automatically and transparently recorded 'focus change' record includes the following fields: 1) a timestamp, 2) a reference to the application running on the window having the focus, 3) a user defined categorization. When the user explicitly forces the system to make a landmark, the previously described record is extended by a snapshot of the focused window, and other data provided by the user (e.g. a comment).

The two streams of events shown side-by-side in Figure 2 give an idea of how the context-aggregator works. Nevertheless, the true potential of merged streams is only revealed by powerful visualizations.

5.2. Visualizing the past activities: the context-view

The context-view provides a visual front-end to the context-aggregator's repository. From the user's point of view, the context-view acts as a map-like resource to the user's past activities.

In general, different usage data visualization techniques allow researchers to exploit their innate visual analysis capabilities to analyze results [18]. These techniques have traditionally been used by the usability experts and not by the end-users. Yet, as Simon [33] argued, users can be considered as experts when reflecting on

their activities, thus it might turn out to be beneficial to provide similar visualization techniques in an interactive form for the end-users.

For visualizing events that span over substantially long period of time, a calendar-type of visualization is useful [13]. Our concept of *cumulative history of events* refers to the idea of categorizing and visualizing the usage data without paying attention to the micro-level temporal dependencies. This is closely related to 'history-enriched digital objects' [19] – an approach that is based on the idea that real world objects accumulate signs of "wear" when used and that scuffing informs future usage. Hill et al. [20] illustrated the idea in the domain of document processing: every episode of reading and editing a document leaves its mark to the document.

Figure 3 presents an example of how to effectively visualize a cumulative history of events. A view of John Smith's cumulative history of events during one week is presented. Besides giving an overview and useful insights of the workload of the week, the cumulative history of events can facilitate the search of some particular task or activity. For example, John recalls that he had been working almost a day on the component A during the week. So, most probably it was Tuesday, when he had made the major changes there.

Figure 3 illustrates how low-frequency events have been generated from a number of high-frequency events. The events range from high-frequency micro-level interactions that are short-lived and mostly synchronous, to high-level interactions that are of low-frequency and to a large extent asynchronous [18; 32]. The view in Figure 3 can thus be seen as a 'plan' that was never explicitly made but, nevertheless, materialized through the user's activities. The interconnectedness of plans and actions is related to Suchman's [35] idea of plans as representations of situated actions.

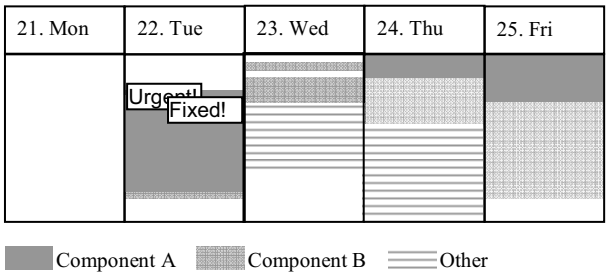

Figure 3: An overview of John Smith's activities from the past week.

The calendar view in Figure 3 shows a discontinuous, low-granularity view of actions over a substantially long period of time. It provides an overview to thematic changes in the user's long term activities. The legends 'Urgent!' and 'Fixed!' in Figure 3 represent landmarks of actions, i.e. stages that the user has explicitly considered significant with respect to ongoing work.

Figure 4 shows the zoomed-in view of the activities on Tuesday 10 – 11 am. It represents the same landmarks of actions as in the calendar view. Thus, these landmarks of actions act as 'hooks' between long-term, low-frequency activities and micro-level activities. The black bars in Figure 4 indicate constant focus on a specific application. Labels above the bars refer to a particular content (that was disambiguated from the filename or application window in the context-aggregator component). These labels act as reference points to corresponding contents. Landmark labels (e.g. 'Urgent!') refer to supplemental information provided by the context-aggregator (e.g. a snapshot of the window).

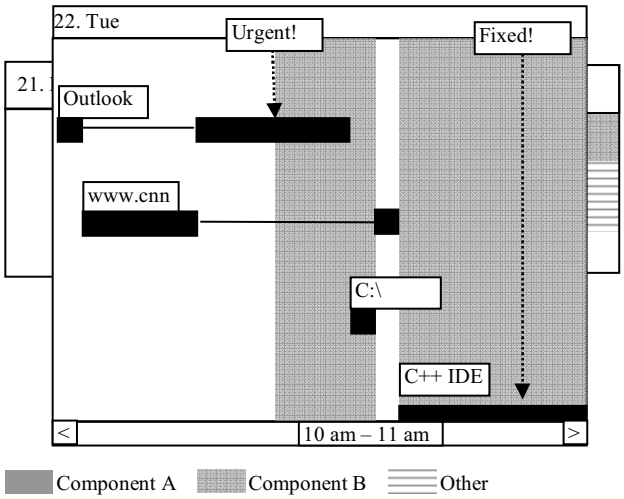

Figure 4: Activities spanning over an hour of time.

Altogether, the example of context-view (Figures 3 and 4) acts as an interactive visualization to the user's past activities in the desktop environment. The context-view described here facilitates getting a visual overview of activities emphasizing the landmarks of actions. It also helps the retrieval of supplemental information.

The view in Figure 4 adopts techniques from usability research where results of performing selection or abstraction on an event stream are visualized using a timeline [18]. The difference in our case is that the information is used by the user of the system instead of the system's designer or researcher.

6. Related approaches

There is a collection of related studies that have addressed the importance of temporal information and the user's activity-history in managing personal or collaborative information space. However, none of them has emphasized the user's subjective evaluation of the significance of ongoing activity (i.e. landmarks of actions) the same way that we have.

Fenstermacher and Ginsburg [9] have presented a framework that monitors higher level user-events to learn how people access, create, and modify information. Despite the surface-level similarity to our approach, their emphasis is on organizational level benefits instead of personal information management: enabling feedback on usage of information sources, getting usage-data of frequently connected series of activities for user-interface designers, etc. Their framework takes advantage of Windows Component Object Model (COM) and is, thus, restricted to Windows environment.

In some suggested solutions, the role of time in organizing information has been emphasized. For example, Rekimoto's [30] time-centric approach integrates historical perspective to the user's personal information space: the user can 'travel' in time and the whole environment changes to the time in question. Lifestreams [11] provides a storage model of its own for organizing data time-wise in a personal workspace. These solutions differ from ours in trying to extend or change the desktop metaphor as a whole. Also Edwards and Mynatt [7] and Hayashi et al. [17] have addressed the importance of time related information. Edwards and Mynatt's Timewarp-toolkit [7] makes it possible to build applications that can handle documents with multiple timelines. Hayashi et al. [17] have focused on web-documents and how to provide an activity-based perspective on them to help knowledge workers. These approaches are directed more towards supporting collaboration. Also, they try to provide new data infrastructures and new applications instead of building on the existing applications in contrast to our approach.

TimeSpace [23] is an example of how to use temporal information in personal information space management and augment the existing desktop environment. It provides activity-oriented document-centric virtual workspaces, within which temporal layouts of information items are represented for retrieval, and shows an overview of activities. Nevertheless, it does not take into account user's experience on the significance of the different activities.

There are some intuitive solutions for visualizing documents or records over time. ThemeRiver [16], for example, can be used for visualizing thematic changes in a large document collection over time. Especially relevant from our point of view is the LifeLines [28]: it introduces an intuitive and general technique for representing a variety of personal history records – similar to our context-view.

7. Discussion

"While most people tacitly understand what context is, they find it hard to elucidate" [5, p. 4]. In this respect, desktop environment makes no exception. We argue that contextual issues concerning the desktop environment are far from trivial and the research efforts directed towards context-aware computing could provide a fruitful basis for designing desktop applications as well. The debate on defining the context in context-aware computing can, on one hand, increase the understanding of the individual user's working context in desktop environments. On the other hand, the critique on different definitions of context in context-aware computing can point out tacit assumptions that have steered the design of desktop applications. As Chalmers [3] has asserted, designers do influence and constrain the predetermination of meaning and the contextualization of a design.

We have come up with an imaginary use case that we have analysed in order to depict and illustrate some problems the user faces when disassembling his working context. Applying this kind of artificial use case, on one hand, facilitates highlighting some relevant concerns and, on the other hand, creates a substantial risk towards purpose-orientation, i.e., building the use case to fit with the argument. One of the main assumptions delimiting the scope of the article is that of desktop environment being a central part of knowledge-worker's working environment - and this very same assumption or premise is behind the imaginary use case. Despite being situated in a specific working environment (i.e. one of software developer's) we see that the use case promotes a set of general aspects of desktop working (e.g. interruptions, long-term activities and shifts between the activities). Thus, we argue that it can serve as an 'archetype' for desktop working. Reflecting the issues from the analysis of the use case on a study of information intensive work (e.g. [8]) reveals that our list of features is well in line with the issues raised in everyday computing, showing the need for managing temporal dependencies between one's personal activities. In order to bring this rather abstract design requirement into operational level, we have come up with the concept of *interoperationality*. We use the concept of interoperationality in contrast to a more technology-oriented concept of *interoperability*.

In our approach, the starting point has been in personal work and in desktop environment. This involves a technical challenge of finding the least common denominator in these kinds of circumstances. In order to tackle this challenge and, at the same time, to emphasize the nature of desktop work from the user's point of view, we have addressed the unstructured shifts between different applications and contents in the course of time. There is no need to change the way people work. What we offer instead is a new "map-like resource" on the person's working

history. From the systems design point of view, this, on one hand, suggests a new class of applications supporting user's awareness of the desktop working environment as a whole and, on the other hand, helps in charting resources and means the desktop environment provides for the implementation of such applications.

We have proposed an abstract application model that conceptualizes the management of temporal dependencies between one's personal activities. The model comprises both system level and user interface level descriptions. The idea of landmark of action addresses an important issue that has not been discussed before. Namely, that the user can have tacit knowledge of that the thing he/she is focused on at a certain moment is important and should be captured for a later use either for the person herself or for organizational purposes. The application we have modelled emphasizes proactivity of the user, i.e. it is up to the user to express when something important is going on. However, the question of what remains: what is the thing he or she is focused on? The abstract model we have proposed (Figure 1) does not take a fixed stance on this issue, but we have offered a rather straightforward and application independent proposal for a solution to this problem. There are some limitations in this generic solution: for example, not all applications store and retrieve their data from the filesystem. However, it is still possible to attach some other data to the landmark (window capture, data from the clipboard etc.)

We believe that the interoperational approach for 'presenting the past' of user's own work would make the user an expert when reflecting on his own work. The examples and related studies demonstrate a need for such support. The system is still a paper prototype, but the model drafts some specifications for the implementation.

8. Conclusion

In this paper we have focused on individual user, personal knowledge management, and intrapersonal aspects of the working context. However, we have also taken into consideration that intersubjective aspects create different demands and restrictions on intrasubjective working environment. We have considered the debates on CSCW and on context-aware computing and taken a rather practical stance in our work with respect to the expansion of digitalization. Although almost all contemporary working environments are partially digitalized, we share Kirsh's [22] assumption about the limits of the extent to which a physical space can be digitally augmented and reshaped. Rhetorically, one can ask how the contemporary digitalized working environment can meet the needs of one's work context.

The articulation of one's working context enables personal knowledge management and, consequently, sharing the knowledge with others. When Ehrlich and Cash [8] studied corporate librarians, they found that the ability to retrace the information retrieval process of the individual information seekers would promote organisational knowing. This is in line with McDermott's [24] characterization of knowing and knowledge: knowing is in actions of individuals. Thus, supporting personal information management by articulating personal knowledge through automated capturing of activities would enable the knowledge to be shared among other people within the organisation, when needed and wanted.

A very special feature of the solution described in the article is that it will provide valuable usage data of itself and the whole desktop environment as a side-product through its usage. This feature certainly is a sword of Damocles: on one hand, the application can be helpful not only for the user and for researchers, on the other hand, the application can be misused and the user's privacy infringed. Will the organization monitor users' activities even more than is the case currently if retrospective information on these activities becomes more easily available?

9. Acknowledgements

We thank Minna Koskinen and anonymous reviewers for constructive comments.

10. References

[1] Abowd, G.D. and Mynatt, E.D. (2000) 'Charting Past, Present, and Future Research in Ubiquitous Computing', *ACM Transactions on Computer-Human Interaction,* 7(1), 29-58.

[2] Bowers, J., O'Brien, J. and Pycock, J. (1996) 'Practically accomplishing immersion: cooperation in and for virtual environments', in the *Proceedings of CSCW '96.* ACM, 380-389.

[3] Chalmers, M. (2004) 'A Historical View of Context', *CSCW Journal,* 13(3), 223-247.

[4] Dey, A.K., Abowd, G.D. and Salber, D. (2001) 'A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications', *Human-Computer Interaction,* 16(2-4), 97-166.

[5] Dey, A.K. (2001) 'Understanding and using Context', *Personal and Ubiquitous Computing,* 5(1), 4-7.

[6] Dourish, P. (2004) 'What we Talk about when we Talk about Context', *Personal and Ubiquitous Computing,* 8(1), 19-30.

[7] Edwards, W.K. and Mynatt, E.D. (1997) 'Timewarp: techniques for autonomous collaboration', in the *Proceedings of CHI '97.* ACM, 218-225.

[8] Ehrlich, K. and Cash, D. (1999) 'The Invisible World of Intermediaries: A Cautionary Tale', *CSCW Journal,* 8(1-2), 147-167.

[9] Fenstermacher, K.D. and Ginsburg, M. (2002) 'A Lightweight Framework for Cross-Application User Monitoring', *Computer*, 35(3), 51-59.

[10] Fischer, G. (2001) 'Articulating the Task at Hand and Making Information Relevant to it', *Human-Computer Interaction,* 16(2-4), 243-256.

[11] Freeman, E. and Gelernter, D. (1996) 'Lifestreams: A Storage Model for Personal Data', *SIGMOD Rec.,* 25(1), 80-86.

[12] Goldberg, A. (1995) 'Why Smalltalk?', *CACM,* 38(10), 105-107.

[13] Gray, M., Badre, A. and Guzdial, M. (1996) 'Visualizing usability log data', in the *Proceedings of INFOVIS '96.* IEEE, 93-98.

[14] Grudin, J. (2001) 'Desituating Action: Digital Representation of Context', *Human-Computer Interaction,* 16(2-4), 269-18.

[15] Harper, R.P. (1998) *Inside the IMF: An Ethnography of Documents, Technology, and Organizational Action*, Academic Press, Inc.

[16] Havre, S., Hetzler, B. and Nowell, L. (2000) 'ThemeRiver: Visualizing Theme Changes over Time', in the *Proceedings of INFOVIS '00.* IEEE, 115-123.

[17] Hayashi, K., Nomura, T., Hazama, T., Takeoka, M., Hashimoto, S. and Gumundson, S. (1998) 'Temporally threaded workspace: a model for providing activity-based perspectives on document spaces', in the *Proceedings of HYPERTEXT '98.* ACM, 87-96.

[18] Hilbert, D.M. and Redmiles, D.F. (2000) 'Extracting Usability Information from User Interface Events', *ACM Computing Surveys,* 32(4), 384-421.

[19] Hill, W. and Hollan, J. (1994) 'History-Enriched Digital Objects: Prototypes and Policy Issues', *The Information Society,* 10(2), 139-145.

[20] Hill, W.C., Hollan, J.D., Wroblewski, D. and McCandless, T. (1992) 'Edit wear and read wear', in the *Proceedings of CHI '92.* ACM, 3-9.

[21] Hughes, J., O'Brien, J., Randall, D., Rodden, T., Rouncefield, M. and Tolmie, P. (1999) 'Getting to know the 'customer in the machine'', in the *Proceedings of GROUP'99.* ACM, 30-39.

[22] Kirsh, D. (2001) 'The Context of Work', *Human-Computer Interaction,* 16(2-4), 305-322.

[23] Krishnan, A. and Jones, S. (2005) 'TimeSpace: Activity-Based Temporal Visualisation of Personal Information Spaces', *Personal and Ubiquitous Computing,* 9(1), 46-65.

[24] McDermott, R. (1999) 'Why Information Technology Inspired But Cannot Deliver Knowledge Management', *California Management Review*, 41(4), 103-117.

[25] O'Conaill, B. and Frohlich, D. (1995) 'Timespace in the workplace: dealing with interruptions', in the *Proceedings of CHI '95*. ACM, 262-263.

[26] Ouksel, A.M. and Sheth, A. (1999) 'Semantic Interoperability in Global Information Systems', *SIGMOD Rec*., 28(1), 5-12.

[27] Pekkola, S. (2003) 'Designed for unanticipated use: common artefacts as design principle for CSCW applications', in the *Proceedings of the GROUP '03*. ACM, 359-368.

[28] Plaisant, C., Milash, B., Rose, A., Widoff, S. and Shneiderman, B. (1996) 'LifeLines: visualizing personal histories', in the *Proceedings of CHI '96.* ACM, 221-227.

[29] Reenskaug, Trygve M. H. (2003) The Model-View-Controller (MVC), Its Past and Present, in the *JavaZone 2003 Conference held in Oslo, Norway.* Available on-line at http://www.java.no/web/moter/javazone03/presentations/TrygveReenskaug/MVC_pattern.pdf.

[30] Rekimoto, J. (1999) 'Time-machine computing: a time-centric approach for the information environment', in the *Proceedings of UIST '99*. ACM, 45-54.

[31] Robinson, M., Kovalainen, M., Auramäki, E. (2000) 'Diary as Dialogue in Papermill Process Control', *CACM,* 43(1), 65-70.

[32] Sanderson, P.M. and Fisher, C. (1994) 'Exploratory Sequential Data Analysis: Foundations', *Human-Computer Interaction,* 16(3-4), 251-317.

[33] Simon, H.A. (1974) *The Sciences of the Artificial,* MIT Press.

[34] Sowa, J.F. (2000) *Knowledge Representation: Logical, Philosophical and Computational Foundations*, Brooks/Cole Publishing Co.

[35] Suchman, L.A. (1987) *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge University Press.

[36] Suchman, L.A. (1983) 'Office Procedure as Practical Action: Models of Work and System Design', *ACM Transactions on Information Systems,* 1(4), 320-328.

[37] Terveen, L. (1995) 'Overview of Human-Computer Collaboration', *Knowledge-Based Systems,* 8(2-3), 67-81.

[38] Winograd, T. (2001) 'Architectures for Context', *Human-Computer Interaction,* 16(2-4), 401-419.