Adapting the Multi-Desktop Paradigm Towards a Multi-Context Interface

Sven Schwarz, Malte Kiesel, and Ludger van Elst

German Research Center for Artificial Intelligence DFKI GmbH,
Knowledge Management Department
Trippstadter Straße 122, 67663 Kaiserslautern, Germany

{firstname.lastname}@dfki.de

Abstract. Knowledge workers are typically processing multiple tasks in parallel. As humans can concentrate and work on but one task at a time, users have to switch between them. The required task switching causes a high cognitive load and decreases efficiency. To reduce the costs of these task switches, we have to find ways to cope with a multitude of dormant tasks the user might switch to again. The challenges here are supporting the user to resume work on a (former) task. In this work, we employ the known multi-desktop paradigm and extend it to a task-switching interface. Each of the multiple desktops is identified with one individual *context* of the user. Each virtual desktop consists of a different set of applications with open documents and their positioning on screen. This helps the user to separate individual tasks and allows easy recovering when switching back.
As we explicitly want to allow switching back to finished tasks in order to reuse the knowledge captured there, the multi-desktop paradigm has to be extended to cope with a multitude of desktops (hundreds rather than four or six). This also requires a better user interface for discovering the right desktop when switching back to a recently paused task.

1 Working on multiple tasks "in parallel"

Knowledge workers are typically embedded in a multi-tasking environment: Nowadays most of the workers are working for multiple projects or enacting multiple roles for the company. But even one single project often splits into various separable (sub-)tasks.

In this multi-tasking environment, the work can not be done in a batch-job style. Rathermore, the user is often pressed to process them *interleaved*. While humans can cope well with multi-tasking of simple every-day tasks like, for instance, jogging in the park, singing, looking at flowers, and checking the time using the watch, humans can only *concentrate* on one single task that needs the user's full attention [2, 4]. So, frequent switching back and forth between the tasks is done—sometimes these switches are done intentionally, sometimes due to interruptions (e. g., by other workers, by incoming mail, etc.). According to user studies [6] a knowledge worker's job is interrupted about four times per hour. The authors define *interruption* as follows:

> "We defined an interruption to be a synchronous interaction which was not initiated by the subject, was unscheduled and resulted in the recipient discontinuing their current activity."

Switching from the current task to an older task leads to temporary confusion of the user and costs at least some time getting back to the point where he recently stopped working on that old task. He has to remember the last status, the last actions, the imminent (sub-)goal, etc., as well as rearrange the workplace (find and reopen relevant information items...) to suit the new task—not to mention the cost of tidying up (closing) and storing the old workplace configuration (e. g., setting bookmarks), that is, important open documents, scrolled to currently viewed/worked on passages, etc. A user study [2] has shown that there is a need to support frequent task switching. Especially switching back to a former task should be assisted:

> "The key findings gleaned from the diary study, as well as explicit comments from participants, shaped our pursuit of designs for user interface tools that might better assist users with task switching. The results and comments especially call out the need for software support to ease the challenge of switching back to all projects, but especially to recovering long-lived projects after interruptions."

In the following, we take a look at existing technologies that assist the user during task switching and highlight points that might be enhanced using the context notion.

1.1 Supporting multiple tasks using virtual desktops

Desktop extensions that allow multiple (virtual) desktops to be used and to switch between them are quite common. An analysis of virtual desktop usage [7] has shown that many people partition their computer work using a multi-desktop tool. Traditional multi-desktop tools provide a handful of "virtual desktops" which people typically use to separate different computer tasks. For example, the first desktop is used to read and write business emails, the second desktop is used for software development, the third desktop is used for an inquiry along with creating slides of its results, and the fourth desktop is used for private mail and web browsing.

Typically, it is up to the user to decide what way of using desktops is appropriate. Some users might decide to use one desktop per application; others might choose to use one desktop for one task (i.e., use a mix of applications per desktop). We focus on the latter case. Here, each desktop is identified with one individual context of the user. Every desktop consists of a different set of applications, open documents, and their positioning on screen. This way, using multiple desktops instead of one desktop reduces actions like closing and re-opening windows just for the sake of keeping the workplace clean. This is analogous to using multiple *physical* desks in the real world: papers and material can stay on the desks—there is no need to tidy up the desk before working on a new (or another)

task if you can simply use another desk for the other task. By removing these administrative actions the worker can concentrate on the work itself.

However, the number of virtual desktops typically is limited to relatively few—otherwise selecting the appropriate virtual desktop becomes tedious. Since in the example above desktops are identified with contexts, we propose to add facilities to search for contexts in order to switch desktops.

1.2 Using desktop search to rediscover old tasks

Desktop search has become a well-known feature in the last years. On the desktop, the typical use case of search is to *re*discover documents[1] that have already been read in an older context. One typical workflow here is to (re)start working on a task, using desktop search to rediscover documents, rearranging the workplace (along with viewing applications) to suit the task at hand, and to start processing documents. It is important to note here that in this example, the user effectively wants to reactivate an old task typically—or at least a task that is very similar to the old one the document he searched for was used in.

In this example, using desktop search is merely a kludge—desktop search is used because the notion of documents is clear and desktop search is widely available. Here, the user actually did not want to search for a single document but wanted to (re)create a workplace configuration—of which the opened documents are only a part of. We propose to extend desktop search to allow searching for (task) contexts then, using documents found in keyword search as pointers to potentially interesting contexts and workplace configurations.

2 Multi-desktop paradigm extensions

While traditional multi-desktop frameworks provide only a small number of desktops, people tend to use them to separate applications or classes of applications rather than their manyfold concrete tasks. This means, each desktop is used as some kind of software environment to solve generic tasks rather than to support concrete contexts. For example: Writing and reading emails is used in a lot of the user's contexts, but still the email client is "hosted" on one single desktop.

Our approach of adopting the multi-desktop paradigm emerged from the idea to assist with the more technical aspects of task switching, namely presenting the right applications and documents for the task at hand. However, the traditional multi-desktop metaphor is not yet sufficient to handle the whole set of a user's different contexts. In the rest of this section we present the respective requirements together with envisioned extensions.

2.1 Large number of desktops

We envision supporting the user with one desktop for each of his tasks. This means, whenever the user starts off a new task, a new desktop is created as well.

[1] Note that desktop search differs from web search here: On the web, search is far more often used to find previously *unknown* information.

To be more precise, it is actually the other way round: As the multi-desktop metaphor is the envisioned interface to support task-switching, the user starts off a new task by explicitly creating (and switching to) a new desktop. Hence, the multi-desktop paradigm has to be adapted to cope with a large number of desktops. Moreover, the set of desktops is not stable but continuously increases as new user tasks emerge.

2.2 Persistency of desktops

The multi-desktop paradigm has to be extended to allow persistency of the desktops. First of all, even in the future the number of open applications and application windows will be restricted by processor power and memory. Handling hundreds of desktops with at least one open application (window) per desktop will soon reduce the computer's performance. But besides this minor performance problem the computer will have to be restarted from time to time.

In order to ensure long-term conservation and recovery of a desktop's state, technical issues have to be solved: The set of open application windows, their geometry on screen, and their internal states (especially opened documents) describe a desktop's state. So, a mechanism and user interface is needed to retrieve, store, and recover desktop states.

The current implementation can not guarantee these persistency issues for every application. Instead, we have focused on conservation and recovery of file explorer windows (open file directories) and web browser windows using Mozilla Firefox. Open file explorer windows can be handled quite easily while restoring a browser window required the development of a special recovery plugin for Mozilla Firefox. This works quite nicely in our prototype, including recovery of scrolling positions of previously opened web pages.

We currently neglect the problem arising from multiple versions of a document to be shown in an application window. At the moment we identify a document only via its location on the hard disk or on the web. Due to the lack of a universal, time-oriented versioning system, we store and restore a document using its location only. Hence, when restoring a desktop from the past chances are that the recovering process reveals documents with newer versions than originally present. However, before solving this versioning issue, we first deal with the methodological aspects of the extended multi-desktop paradigm: conserving and restoring desktops as well as respective user interfaces.

2.3 Relating desktops with the user's world

In the Mymory project[2], a so-called PIMO (*Personal Information Model* [9]) provides a vocabulary for describing information elements on an individual desktop, thereby comprehensively reflecting a user's personal view on his information landscape. Similar to topic maps, the PIMO defines relatively informal concepts

[2] `http://www.dfki.de/mymory/`

and relationships (*related-to* etc.) as well as more formal aspects using the expressivity of RDF/S (subclass and subproperty relationships). The Mymory PIMO offers rather general concepts of knowledge work (*Person*, *Organization*, *Location*, *Document*, *Topic*, etc.). A user typically extends this upper model with more specific group or personal concepts (e. g., concrete project types or organizational structures). Most important are the user's instances of these concepts (e.g., concrete persons, or topics).

PIMO classes and instances can both be used for annotations—hence we define and use the term PIMO "concept" for both: a class or an instance. In contrast to standard Web 2.0 style tagging that uses binary relations, we use a weight between `0.0` and `1.0` defining to what extend this annotation holds for each concept annotated with a desktop. A strong relevance of this concept for the desktop is indicated by `1.0`; a lower value defines a weaker relevance. On the one hand, these weight values can be set and adjusted by the user. One the other hand, they are estimated and adapted by an automatic context-awareness component (see section 2.4). These weighted annotations are used in the user interface: For each desktop its weighted concept annotations are presented in form of a "tag cloud" as it is used in typical Web 2.0 applications (see figure 2).

For each desktop we store its individual set of weighted annotations in a vector, holding the relevance value for each PIMO concept. We use a dynamic vector for this, such that a vector's dimension increases whenever a relevance to new PIMO concept emerges. Figure 1 shows how the concepts in the PIMO are mapped to an index used in a PIMO vector describing a user's context. We end up having such a vector for each desktop, which enables simple vector arithmetics to find, organize, and relate desktops. For example, a commonly used cosine measure is applied as a similarity measure between desktops.

Annotating desktops with these concepts allows to search for and to filter desktops with these concepts. Hence, a relevant desktop to switch to can be found quite easily. Moreover, these concept annotations classify the desktops and enable a similarity measure between desktops. This is used to display "similar" desktops to the current desktop, which facilitates the reuse of information buried in a desktop with similar classification (similar annotations). The similarity measure can also be used to cluster the desktops or to generate and render a 2D map of the desktops (self-organizing feature map).

2.4 Context-awareness automatically adapts desktop annotations

These annotations are provided automatically by the Mymory context service. Automatic user observation generates a continuous stream of contextual evidence which is then fed into a context elicitation framework. That way, the user's context is captured without needed intervention of the user. The `User Observation Hub`[3] is an open source Java project responsible for the gathering and distribution of user observation data. Technically, this is achieved by

[3] `http://usercontext.opendfki.de/wiki/UserObservationHub`

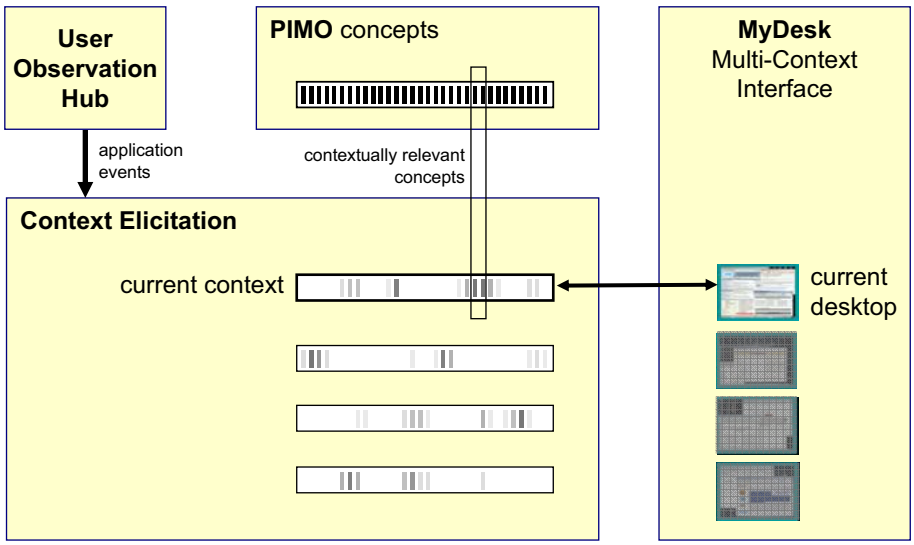

Fig. 1. Automatic user observation feeds the context elicitation. As desktops are described with these contexts, the desktops adapt according to the user's behavior.

observing the usage of a number of applications such as the web browser and editor using plug-ins and keeping track of the events within these applications. The Mymory context service makes use of these events and the content displayed in the applications and matches them against a number of rules that are partially autogenerated from the user's PIMO. That way the user's context model is filled with weighted estimates of PIMO concepts being contextually relevant. Identifying one desktop with one context leads to automatic contextual annotations of each desktop as described in section 2.3. A detailed description of the user observation and context elicitation framework is beyond the scope of this paper. See [10–12] for an overview on modeling, using, and accessing user context for knowledge management scenarios.

Figure 1 shows the interaction of the context-related system components: The user observation on the top-left sends application events to the context elicitation with which adapts contextually relevant parts the current context. The concepts used for the context elicitation and annotation are provided by the user's own, individual PIMO which you can see at the top of the diagram. This context is captured using the vector model described in 2.3, that is, for every context there exists such a vector as it can be seen in middle of the diagram. As we describe each desktop with its *assigned* context, a desktop's annotations adapt automatically as soon as its context changes. The MyDesk interface on the right side manages and visualizes the whole set of desktops and controls which desktop (and hence which context) is active—see the following section for more details.

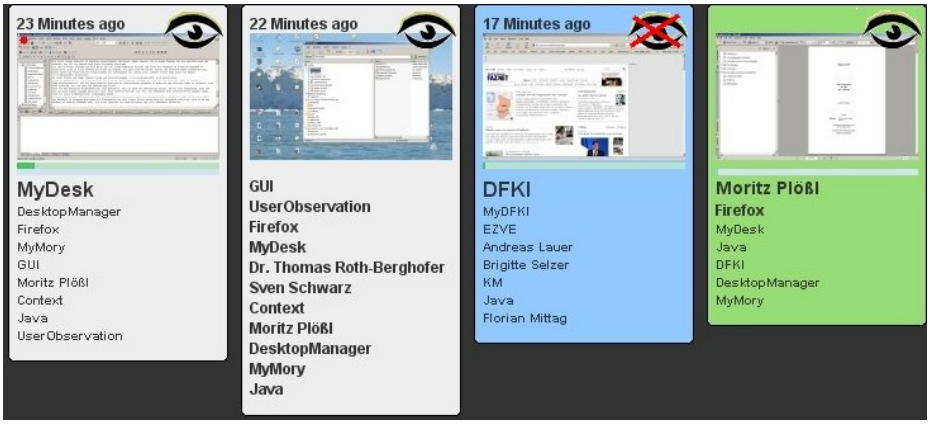

Fig. 2. A timeline view displays the most recent desktops via a most recent screenshot and a "tag cloud" of the most relevant concepts.

3 Towards an implementation of a multi-context interface

`MyDesk`[4] is an open source implementation of the adapted multi-desktop metaphor described in section 1.1. The main parts of MyDesk (i.e., the user interface and the desktop-switching API) are implemented in Java and, hence, platform independent. Only a small part (the native code needed to trigger the desktop switching) is done using C# and .NET (for Windows) and C (Unix).

While the user interface is still under development, prototypical components have already been implemented. In figure 2 a timeline view displays property cards for the most recent desktops. Each card shows a screenshot of the last desktop state (open windows) together with a "tag cloud" showing the most relevant concepts describing the context of this desktop. As typical for tag clouds, the degree of relevancy of one concept is shown using the font size. For our approach the degree of relevancy for one desktop coincides with the activation level of the concept for the corresponding user context. Currently, the concepts are ordered according to relevancy. However, as this makes the used font size redundant we will investigate different ordering strategies (e.g., alphabetical) in the future. The state of the eye in the top-right corner of such a desktop card indicates whether this desktop will be observed or not. The user can switch user observation on/off at any time—this configuration is desktop-specific.

Figure 3 shows a mockup of an envisioned clustering and search interface. A two dimensional map groups together similar or interrelated desktops. As the context representation used for the desktops comes with a similarity measure, automatic clustering of desktops can be accomplished, e.g., via self-organizing feature map techniques. Additionally, besides these annotation-based relations, the occurred desktop switches convey information about temporal correlations.

[4] `http://mydesk.opendfki.de/`

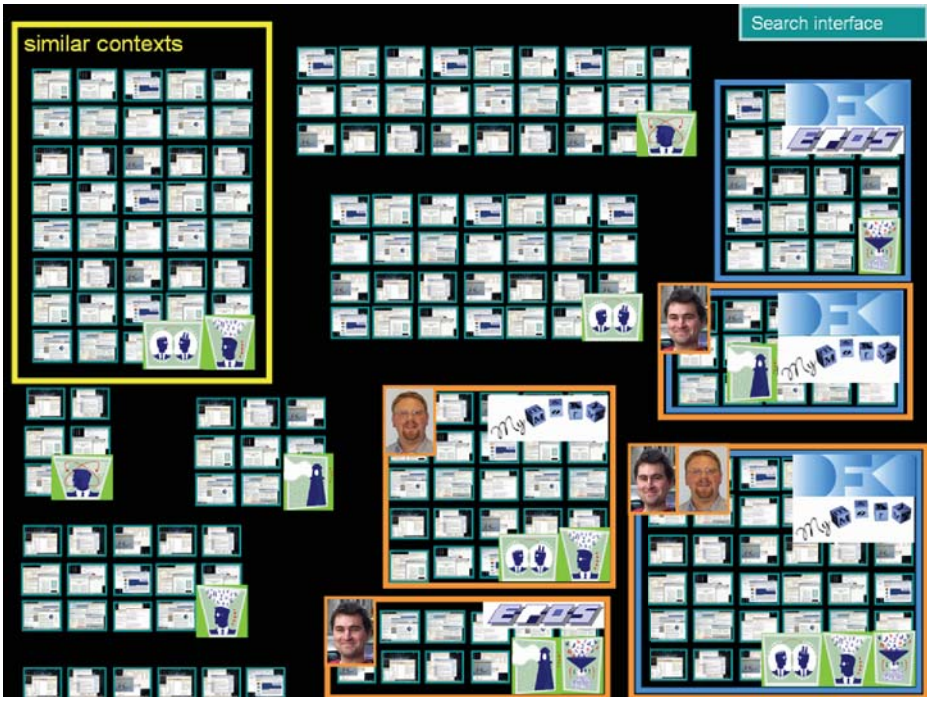

Fig. 3. A two dimensional map groups together interrelated desktops.

Assuming (potential) causal relations for some of these temporal relations we can use these for visualization or even clustering, too. However, we believe that the user may also want to locate the desktops himself, according to his individual feeling with respect to the relationships between desktops.

A text field in the top-right corner allows successive filtering. An example result after entering two search terms is shown in figure 4. Only the desktops relevant to the search terms are visible, partial relevance can be recognized by a smaller icon size. The important point here is, that the desktops stay at the same position and irrelevant desktops are hidden. A one dimensional result list with desktops re-ordered according to their relevance (with respect to the search) risks the user getting "lost in desktop space".

We believe that a user will need different visualization and search metaphors to actually find the relevant desktop. Hence, the user should be free to choose and switch the visualization. However, selected desktops should still be focused after changing the visualization. For example: Imagine the user selects a group of desktops in the clustering/2D map view and then switches to the timeline view to see when each of those desktops has been accessed recently. Then the previously selected desktops will still be selected (and hence highlighted) in the timeline view. He also sees non-highlighted other desktops in between the selected ones.

Fig. 4. Only the desktops relevant to the search terms are visible, partial relevance can be recognized by a smaller icon size.

He could leave the selection as it is, but also decide to unselect some or select others and switch back to the 2D map view to see if and how they interrelate.

The scalability and usability of such a multi-context user interface depends on the available options to visualize, filter, and search for desktops. Moreover, the *combinations* of different filter and visualization metaphors as well as the possibility to *switch* between different metaphors (while keeping the same focus) will play an important role in finding the right desktop to continue work there. To be more precise, displaying all available desktops via a 2D map visualization showing small icons for each desktop is nice to get an *overview*. However, the icons are pretty small and will not be sufficient to decide which one to pick. Besides search and filter mechanisms, the user will feel the need to see more details—e.g., via zoom and pan. Additionally, filtering out "old" desktops (i.e., desktops not visited recently) using a time-slider can reduce the number of desktops to a manageable set.

4 Related Work

One of the first systems working with *multiple* workplaces to support multitasking workers was ROOMS [3, 1]. ROOMS supports new user tasks by allowing the user to create a new "room" (to be used as workspace) for that task. As the rooms metaphor implies, ROOMS enables the user to connect "rooms" via "doors". Assuming meaningful placement, the user can walk around (i.e., "browse") related rooms via the user-defined doors. Although the rooms metaphor seems naturally intuitive and easy to understand, the disadvantage is, that rooms are connected and interrelated with doors placed *manually* by the user. Adapting the rooms metaphor to propose creation of meaningful doors or using auto-generated doors could improve the scalability. Moreover, ROOMS does not allow *searching* for rooms or getting an *overview* on the whole set of available rooms.

Task Gallery [8] is another task-oriented virtual desktop manager taking the rooms metaphor to 3D. The idea is to visualize a three dimensional gallery in form of a tube. The user's individual tasks and the corresponding open application windows are visualized as images on the walls, floor, and ceiling. The user can create new tasks there at any time. The 3D presentation allows easy navigation (browsing) between the user's tasks, intuitive clustering (similar, interrelated tasks grouped together), and supports as well as utilizes humans' good spatial cognition abilities to remember a task's (geographical) position in the gallery. However, the user can not interact with these tasks directly at the place where they are visualized. Instead, actual processing takes place only at one designated place: the so-called "stage". Analogously to ROOMS, Task Gallery also does not show (potentially) related task, nor does it provide any means to search for tasks or get an overview on the available tasks.

GroupBar [5] and WindowScape [13] solve the problem of many open windows by creating task-specific groups of windows. Switching a task then reduces to selecting the *group* of windows that corresponds to that task. Only the windows of the selected group are visible, the rest is hidden. The user interfaces with a time line visualization of these window groups, which is easy to use and allows to get an overview on available tasks. However, this approach does not scale for a large number of user tasks (very old tasks will be difficult to find if the user does not remember the last access date). Moreover, similarity or relationships between tasks are not modeled nor exploited to guide the user to relevant tasks to switch to.

Switching *profiles* in everyday applications such as web browsers and chat programs can be seen as similar to switching user context. However, profile handling is quite heavyweight—to switch profiles, applications have to get restarted typically. This is due to the fact that profiles are intended to allow multiple users (or one user in different roles) to access the same application without "polluting" application settings and stored data. Features such as the ability to bookmark all currently opened tabs in Firefox and therefore capturing much of the application's inner state in one bookmark seem to indicate that the need for

user context support has been perceived to some degree in end user applications though.

5 Conclusion

Knowledge workers are embedded in multiple projects. It has been shown that this leads to many interruptions and task switches per hour, which in turn leads to a great loss of efficiency. These task switches cannot be removed completely from the daily work. However, we aim at reducing the high cognitive load arising due to interruptions in current work. This is done by supporting the user when adapting the work environment for the next task at hand.

Our approach is to apply and adapt the commonly used multi-desktop paradigm to be used as a multi-context interface. The main goals of the system are

- enabling the user to get an overview over the available contexts,
- showing related contexts for a selected (e.g., the current) context,
- support search for contexts (e.g., search for a context to switch to),
- support resuming a former context by helping the user to remember the last state of this context—this is the main reason for applying the multi-desktop paradigm,
- automatic annotation (representation) of desktops/contexts (only with activated user observation),
- detection of and notification about potential context switches (only with activated user observation).

First tests show that the combination of context-awareness technology and multi-desktop metaphor yields an efficient and easy to use user interface to cope with multiple contexts, especially context switches.

User observation is used to feed an automatic context elicitation service, which automatically maintains context and desktop representations. This enables an unobtrusive way of keeping track of what a desktop is about and allows searching, filtering, and clustering desktops.

The large quantity of contexts per desktop demands some extensions of the multi-desktop metaphor—for example, an arbitrary number of desktops, respectively a more scalable user interface, and the long-term persistency of all desktop states plus corresponding recovering mechanisms. Although the management and conservation/recovering of desktops works quite nicely in our prototype, the user interface is still under development. In particular, some visualization metaphors have not been implemented yet. An evaluation of our approach is planned at the end of 2008.

Acknowledgement

The Mymory project is funded by the Bundesministerium für Bildung und Forschung (Federal Ministry of Education and Research) under grant 01 IW F01.

References

1. Stuart K. Card and Jr. Austin Henderson. A multiple, virtual-workspace interface to support user task switching. In *CHI '87: Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface*, pages 53–59, New York, NY, USA, 1987. ACM.
2. Mary Czerwinski, Eric Horvitz, and Susan Wilhite. A diary study of task switching and interruptions. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 175–182, New York, NY, USA, 2004. ACM.
3. Jr. D. Austin Henderson and Stuart Card. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Trans. Graph.*, 5(3):211–243, 1986.
4. Alan J. Dix et al, editor. *Human-Computer Interaction (2nd Edition)*. Prentice Hall, 1988.
5. V. Kaptelinin and M. P. Czerwinski, editors. *Beyond the Desktop Metaphor Designing Integrated Digital Work Environments*. The MIT Press, 2007.
6. Brid O'Conaill and David Frohlich. Timespace in the workplace: dealing with interruptions. In *CHI '95: Conference companion on Human factors in computing systems*, pages 262–263, New York, NY, USA, 1995. ACM.
7. Meredith Ringel. When one isn't enough: an analysis of virtual desktop usage strategies and their implications for design. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 762–763, New York, NY, USA, 2003. ACM.
8. George Robertson, Maarten van Dantzich, Daniel Robbins, Mary Czerwinski, Ken Hinckley, Kirsten Risden, David Thiel, and Vadim Gorokhovsky. The task gallery: a 3d window manager. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 494–501, New York, NY, USA, 2000. ACM.
9. Leopold Sauermann, Andreas Dengel, Ludger van Elst, Andreas Lauer, Heiko Maus, and Sven Schwarz. Personalization in the epos project. In M. Bouzid and N. Henze, editors, *Proceedings of the International Workshop on Semantic Web Personalization, Budva, Montenegro, June 12, 2006*, pages 42–52, 2006.
10. Sven Schwarz. A context model for personal knowledge management applications. In Th. Roth-Berghofer, St. Schulz, and D. B. Leake, editors, *Modeling and Retrieval of Context, Second International Workshop, MRC 2005, Edinburgh, UK*, volume 3946 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2006.
11. Sven Schwarz and Thomas Roth-Berghofer. Towards goal elicitation by user observation. In A. Hotho and G. Stumme, editors, *Proceedings of the LLWA 2003*, pages 224–228, Karlsruhe, oct 2003. AIFB Karlsruhe, GI.
12. Roza Shkundina and Sven Schwarz. A similarity measure for task contexts. In *Proceedings of the Workshop Similarities - Processes - Workflows in conjunction with the 6th International Conference on Case-Based Reasoning*, 2005.
13. Craig Tashman. Windowscape: a task oriented window manager. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 77–80, New York, NY, USA, 2006. ACM.