Scalable Fabric: Flexible Task Management

George Robertson, Eric Horvitz, Mary Czerwinski,
Patrick Baudisch, Dugald Hutchings, Brian Meyers, Daniel Robbins, and Greg Smith
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
{ggr; horvitz; marycz; baudisch; brianme; dcr; gregsmi}@microsoft.com; hutch@cc.gatech.edu

ABSTRACT
Our studies have shown that as displays become larger, users leave more windows open for easy multitasking. A larger number of windows, however, may increase the time that users spend arranging and switching between tasks. We present *Scalable Fabric*, a task management system designed to address problems with the proliferation of open windows on the PC desktop. Scalable Fabric couples window management with a flexible visual representation to provide a focus-plus-context solution to desktop complexity. Users interact with windows in a central focus region of the display in a normal manner, but when a user moves a window into the periphery, it shrinks in size, getting smaller as it nears the edge of the display. The window "minimize" action is redefined to return the window to its preferred location in the periphery, allowing windows to remain visible when not in use. Windows in the periphery may be grouped together into named tasks, and task switching is accomplished with a single mouse click. The spatial arrangement of tasks leverages human spatial memory to make task switching easier. We review the evolution of Scalable Fabric over three design iterations, including discussion of results from two user studies that were performed to compare the experience with Scalable Fabric to that of the Microsoft Windows XP TaskBar.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *graphical user interfaces, windowing systems*.

General Terms
Management, Design, Experimentation, Human Factors.

Keywords
Task Management, Scaling, Interaction, Spatial Memory.

1. INTRODUCTION
Twenty years ago, Bannon *et al.* [1] observed that information workers often switch between concurrent tasks or activities. In Rooms, Card and Henderson [6] observed that tasks can be supported via the management of "working sets" of windows, in much the same way operating systems manage working sets in memory. Card and Henderson identified desirable properties of task management systems, including: fast task switching, fast task resumption, and easy reacquisition of the cognitive context associated with a task.

Over the two decades since the work of Bannon *et al*., numerous virtual desktop managers have been built and each has exhibited some of these properties. Task management systems typically provide some efficient way of switching from one set of windows and applications to another set, as a basic form of task switching.

Although workers may switch among tasks in a self-guided manner, a significant portion of task switching is caused by external interruptions [12]. Czerwinski, Cutrell, and Horvitz [7][9][[10] have sought to understand the influence of interruptions on task switching for information workers in order to design tools that can assist users to recover from interruptions.

We have also been motivated to re-examine task switching and task management design opportunities in the face of the growing popularity of larger display and multiple monitor configurations. In an informal study at our organization, we found that when users shift to larger display surfaces, they leave more applications running and associated windows open. For example, we observed that single display users tend to keep an average or 4 windows open at once, while dual monitor users keep 12 and triple monitor users keep 18 windows open on average (N=16 users). This significant trend suggests that there is an opportunity for design innovation with windows and task management to make handling larger numbers of concurrent windows, potentially clustered by task, a fundamentally more natural and effective experience.

We have developed a windows management methodology to exploit this opportunity. *Scalable Fabric* is a system designed to assist users manage tasks on the Windows desktop, allocating screen real estate in accordance with a user's attention, using a focus-plus-context display. The periphery of the screen is used to hold scaled down live windows rather than hiding them with traditional windows minimization. In order to facilitate task switching, Scalable Fabric allows users to group collections of windows that are used together. We shall refer to groups of Windows that are used together as *tasks*. We realize that this notion is not isomorphic with all conceptions of computer-centric "tasks," but in our conversations with end users after studies of this topic, this notion appears to resonate easily with their description of their own computer work.

In the remainder of the paper, we will first discuss related research. Then we will describe details of the Scalable Fabric methodology. We present the results of a comparative user study of Scalable Fabric and the Windows TaskBar, and a longitudinal field study of Scalable Fabric. Finally, we discuss project directions and opportunities for future research.

2. RELATED WORK ON TASK MANAGEMENT

The most popular software system for task management is the *virtual desktop manager*. One of the earliest designs exploring a virtual desktop manager was Smalltalk Project Views [13]. *Rooms* [6][15] is probably the most well-known of these kinds of systems. A number of virtual desktop managers are currently available, and are described in [27]. We have not been able to find evidence that these systems have been evaluated in a formal manner. Thus, it is difficult to ascertain how easy they are to use or how well they integrate into real-world settings.

In addition to virtual desktop managers, a number of novel solutions have been proposed, including extending the user's desktop with additional low-resolution screen space [2], employing 3D environments as pursued by the TaskGallery [24] effort, providing a zoomable space as in Pad++ [3], and the use of time as the main axis and organizing principle [22]. Also, tiled window managers [5][26] have been created to address some of these same issues, as well as systems that involve the invocation of bumping processes among windows, to allow a window at focus to push others away [4][16].

We have pursued prototypes of temporal and spatial visualizations of users' daily computing configurations. These designs use lightweight, temporal cues, such as the state of a user's desktop at different times [19]. We have also sought to provide support for task-based visualizations and switching, in a similar vein to the work of Henderson & Card [15], Kaptelinin [17], Macintyre et al. [18] and Robertson et al. [24].

In distinction to the prior work, we have explored designs for virtual desktop organizers that do not replace the entire PC desktop with a new metaphor, but rather occupy the same conceptual and physical space that is already devoted to window management in the Windows OS – namely, the area in the periphery of the display surface. Using these prototypes, we have been performing longitudinal studies on the benefits of temporal and visual cues for enhancing memory about knowledge-based tasks, in order to facilitate task switching. We seek to understand the potential benefits from the use of these systems, and to iterate their design. For example, the Windows XP TaskBar provides "grouping by application" to address the problem of running out of bar space, *e.g.*, all Word windows are grouped together, and all Internet Explorer windows are grouped together. Grouping by application, rather than by task, can create user confusion, as specific windows executing the same application may be conceptually unrelated to each other, and cross-application windows may be used together on one user activity [12].

We also address the challenge of accessibility of windows belonging to different tasks. While virtual desktop managers typically impose strict separation between tasks, we allow users to simultaneously display any subset of windows, even if they should be assigned to different tasks. Rooms' placements mechanism allows a window to appear in multiple virtual desktops, but this requires forethought to set up. The approach in Scalable Fabric is more dynamic and requires no forethought.

In related work, GroupBar [25] addresses these latter issues by evolving the Windows TaskBar to support task groups of windows on a bar, using the same minimized window representation used by TaskBar.

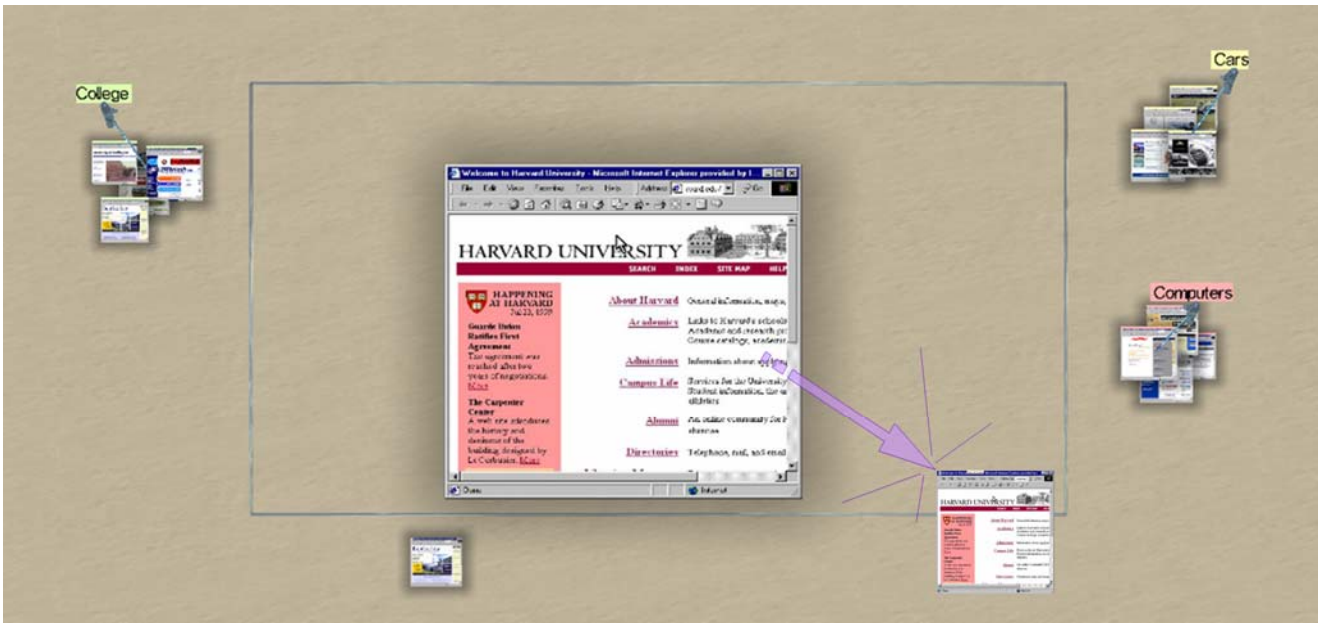
Although GroupBar has most of the properties we were seeking in a task management system, the design does not effectively leverage human spatial and visual recognition memory. We know from user studies on the Data Mountain [23] and Task Gallery [24] that spatial memory works in a virtual environment similarly to the way it works in the physical world, and that user task performance is enhanced, particularly when the task involves retrieving items placed spatially. GroupBar makes limited use of spatial memory by allowing users to create multiple bars. Limitations stem from the bar design, which is linear, list-based, and does not expose much virtual space in which to place tasks.

Scalable Fabric makes use of the periphery of the display for spatial layout of tasks, in addition to leveraging users' efficient visual recognition memory for images [8]. Scalable Fabric allows users to leave windows and clusters of windows open and visible at all times via a process of scaling down and moving the windows and clusters to the periphery. This idea was partially inspired by observations we made with Data Mountain; items toward the back of the Data Mountain take much less space, but are still readily recognizable. It was also inspired by the scaling at the edges of the display in Flatland [21] and by ZoomScapes' location based scaling mechanism [14]. While ZoomScapes is not a task management system, its management of sheets and groups of sheets is similar to Scalable Fabric's management of windows and tasks. We shall review the differences in the next section.

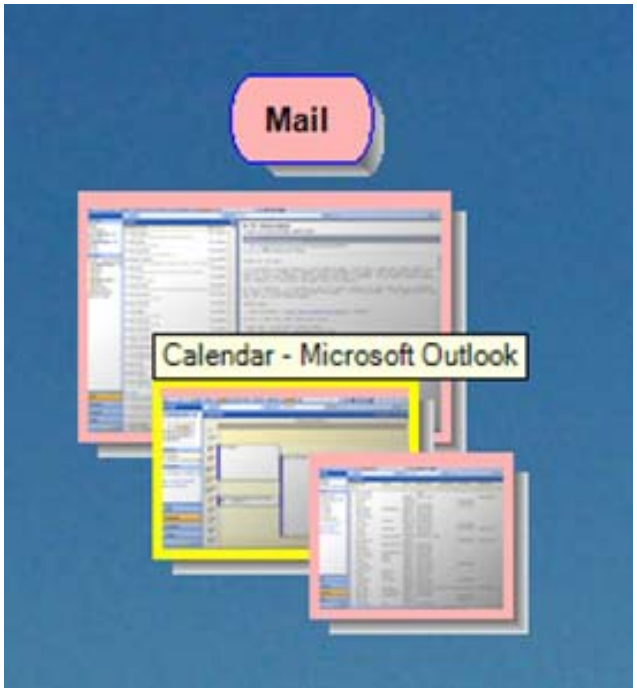
3. SCALABLE FABRIC BASICS

In Scalable Fabric, the user defines a central focus area on the display surface by moving periphery boundary markers to desired locations. In Color Plate 1, these boundary markers are visible, but users usually hide the boundary markers unless they are changing the size or shape of the focus area, in which case the markers serve as resize handles. Within the focus area, windows behave as they normally do in the Windows desktop. The periphery contains windows and collections of windows (or tasks) that are not currently in use, but may be put to use at any moment. Windows in the periphery are smaller so that more tasks can be held there when the user is focusing on something else. With this metaphor, we believe users will rarely need to close or minimize windows in the traditional sense. Users can take advantage of extra screen real estate, especially on larger displays, to allow the peripheral windows to always be visible.

When a user moves a window into the periphery, it shrinks monotonically with distance from the focus-periphery boundary, getting smaller as it nears the edge of the screen. When the user clicks on a window in the periphery, it returns to its last focus position; this is the new "restore" behavior, and is accomplished with a one second animation of the window moving from one location to the other. When the user "minimizes" a window in the focus area, *e.g.,* by clicking the window's 'minimize' button, it returns to its last peripheral position.



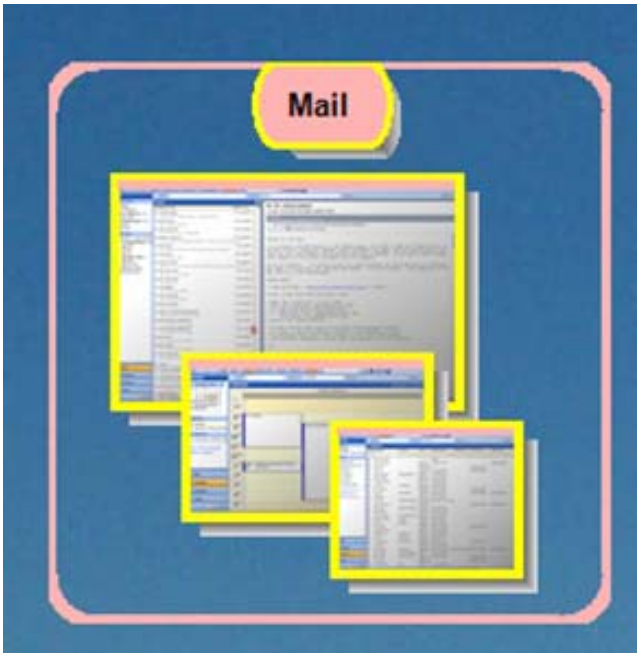
Color Plate 1. Scalable Fabric showing the representation of three tasks as clusters of windows, and a single window being dragged from the focus area into the periphery.



Color Plate 2. Close-up of task (third design).

When a window is moved in the periphery, other windows temporarily move out of the way. This is the occlusion avoidance behavior employed in the Data Mountain [23], and it makes it impossible to obscure one peripheral window with another.

Scalable Fabric uses natural metaphors and gestures that allow users to define, access, and switch among tasks. To define tasks, windows in the periphery are grouped into clusters associated with a colored banner showing which cluster they are in. Moving a window near a cluster marker makes it part of that cluster. When



Color Plate 3. Task highlighting during hover.

clusters are moved around, they avoid each other similar to the way windows avoid one another. The whole point of this behavior is to make it easy for users to construct task clusters by dragging and dropping windows onto groups of windows.

To create a new task, the user simply moves a window near another that is not in a task. The new task is then created implicitly. The user can return later and rename the task. Until the task is named, it is ephemeral. That is, if the last but one window is moved out of an ephemeral task, the task will be un-created (*i.e.*, the task marker will disappear).

Users can use other natural gestures to access and toggle among tasks efficiently. When a user clicks on a task marker, the entire task is selected, restoring its windows to their focus positions. If the user clicks on a task marker when all of its windows are currently in the focus area, each window returns to its peripheral position. If one task is selected and the user clicks on a different task marker, a task switch occurs, *i.e.*, all windows of the current task move to their peripheral positions, and the windows comprising the task being selected in the periphery move to their previous configuration in the focus area.

The user's choice of focus area location and size is influenced by the configuration and capabilities of the physical displays. For example, on a triple-monitor display, some users may prefer to define the central monitor as the focus area, with no upper or lower peripheral regions and the side monitors as the side peripheral regions.

The information in the periphery may be occasionally obscured by open windows (*e.g.,* a maximized window). This can be resolved with two mechanisms. First, any interaction that involves the periphery must make all the periphery windows and task markers visible. Second, there must be some way to make the periphery visible on demand. The solution we have adopted is similar to the TaskBar auto-hide mechanism. Any time the user bumps the cursor into any screen edge, the periphery auto-reveals itself. If the user interacts with any window not in the periphery, the periphery will drop to the bottom of the window z-order.

Scalable Fabric is a focus-plus-context display in the sense that users focusing their attention on a primary task are provided with the context of other work (*i.e.*, competing or potentially related tasks) displayed in the user's periphery.

For moving and scaling windows and groups of windows in Scalable Fabric, we considered findings from ZoomScapes [14]. As windows are rectangles rather than points, it is important to identify the point about which scaling occurs. Like ZoomScapes, Scalable Fabric uses the cursor location (*i.e.*, the drag point) as the scale point. We experimented with several different alternatives, and concur with the earlier work that the cursor position is the most useful scale point.

When moving a group, scaling the windows in the group is not sufficient. Zoomscapes scales the distance between the center of the sheets and the cursor dragging point. In Scalable Fabric, we found a more pleasing effect by scaling the distances from the window centers to the center of the group. That is, as the group gets smaller, the windows move closer together.

When a window moves across the scaling boundary, an abrupt change in scale is disconcerting. Zoomscapes solves this by have a bridge zone where a sharp ramp in scaling is applied. Scalable Fabric uses a different approach, and applies a half-second transition animation to the new scale. This appears to be more graceful than the ramp-zone approach.

4. ITERATIVE DESIGN

We have pursued a process of iterative design for refining and testing versions of Scalable Fabric. To date, we have created three implementations of the system.

The first version of Scalable Fabric was a prototype that worked with images of windows, which allowed us to refine the visual design and interaction behaviors. Color Plate 1 is a screenshot of the first design prototype with the addition of an indication of what it is like to drag a window from the focus area to the periphery. Informal studies were conducted to collect usability issues to drive the second design iteration.

The second design worked with real windows on the Windows Desktop. A user study comparing Scalable Fabric to the Windows XP Taskbar suggested that Scalable Fabric was easily learned and considered valuable by the participants, but several usability issues were noted. A study showed no significant difference in task performance time between the two approaches.

The third version of Scalable Fabric was developed as a set of refinements on the second design. Color Plate 2 shows a close-up of the appearance of windows and task markers, with the cursor hovering over one window to show its title tooltip. Most of the time the task marker appears as displayed in Color Plate 2. However, if the user hovers over the marker or moves a window into the task group, a box appears as rendered in Color Pate 3.

In order to gather further information about how people actually use virtual desktop managers, and to begin to understand in a more detailed manner how Scalable Fabric might be used in real situations, we conducted a second, longitudinal study with 13 participants using their real systems and tasks. Although we confirmed that we had solved many of the usability issues identified in the first study, the second study revealed new opportunities for design iteration. Specifically, as core issues with the design are addressed, system performance and bug fixes have become more important to our end users. This is natural at this stage of the design process, and we will continue to track down any remaining bugs and refine performance as part of future work.

Approximately 200 people have used Scalable Fabric over the last year. The response has generally been quite favorable, with many users continuing to use it. However, several performance and behavior problems have led some people to stop using the prototype after an evaluation period. Some of these problems can be addressed with changes to the current implementation, but others will require rewriting Scalable Fabric as a replacement for the legacy window manager.

5. CONCLUSIONS

Scalable Fabric provides basic task management, using a focus-plus-context spatial metaphor. Windows in a central focus area behave as usual, while windows in the display periphery are scaled down "minimized" windows. By taking less space, the periphery windows can remain open and live. Task switching is accomplished by a single mouse click. Two user studies have provided guidance for several phases of iterative design of Scalable Fabric, and suggest that users prefer this approach to the standard Windows TaskBar. The studies have also identified problems that still need to be addressed. Most of these problems can be attributed to the decision to build Scalable Fabric on top of an existing window manager rather than building it within or replacing the window manager. A future implementation of Scalable Fabric will address these issues.

6. ACKNOWLEDGMENTS

We would like to thank Francois Guimbretiere for discussions about the relationship between Scalable Fabric and Zoomscapes.

7. REFERENCES

[1] Bannon, L., Cypher, A., Greenspan, S., and Monty, M. (1983). Evaluation and analysis of user's activity organization". In Proc. CHI'83 (pp. 54-57). NY: ACM.

[2] Baudisch, P., Good, N., Stewart, P. (2001). Focus plus context screens: combining display technology with visualization techniques. In Proc UIST 2001 (pp. 31-40).

[3] Bederson, B. & Hollan, J. (1994). Pad++: A zooming graphical interface for exploring alternative interface physics. In Proc. UIST'94 (pp. 17-26).

[4] Bell, B. and Feiner, S. (2000). Dynamic space management for user interfaces. In *Proc. UIST'00,* (pp. 238-248).

[5] Bly, S., Rosenberg, J. (1986). A comparison of tiled and overlapping windows. In Proc. CHI '86 (pp. 101-106).

[6] Card, S.K. & Henderson, A.H. Jr. (1987). A multiple, virtual-workspace interface to support user task switching. In Proc. CHI+GI 1987 (pp. 53-59). NY: ACM.

[7] Cutrell, E., Czerwinski, M. & Horvitz, E. (2001). Notification, Disruption and Memory: Effects of Messaging Interruptions on Memory and Performance. In Human-Computer Interaction--Interact '01 (pp. 263-269). IOS Press.

[8] Czerwinski, M., van Dantzich, M., Robertson, G., & Hoffman, H. (1999). The contribution of thumbnail image, mouse-over text and spatial location memory to web page retrieval in 3D. In Proc. Interact 1999 (pp. 163-170), Edinburgh, Scotland, IOS Press.

[9] Czerwinski, M., Cutrell, E. & Horvitz, E. (2000). Instant Messaging and Interruption: Influence of Task Type on Performance. In Proc. OZCHI 2000 (pp. 356-361). Sydney, Australia.

[10] Czerwinski, M., Cutrell, E. & Horvitz, E. (2000b). Instant Messaging: Effects of Relevance and Time. Proc. HCI 2000, Vol. 2, (pp. 71-76). British Computer Society.

[11] Czerwinski, M. & Horvitz, E. (2002). Memory for Daily Computing Events. In Proc. HCI 2002, (pp. 230-245).

[12] Czerwinski, M., Horvitz, E. & Wilhite, S. (2004). A diary study of task switching and interruptions. To appear in Proc. CHI 2004, ACM press.

[13] Goldberg, A. (1983). Smalltalk-80. NY: Addison-Wesley.

[14] Guimbretiere, F., Stone, M., and Winograd, T. (2001). Fluid interaction with high-resolution wall-size displays. In Proc.UIST'01, (pp. 21-30). NY: ACM.

[15] Henderson, D. A. Jr., Card, S.K. (1987). Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. ACM Transactions on Graphics, 5 (3), 211-243.

[16] Kandogan, E. and Shneiderman, B. (1997). Elastic Windows: evaluation of multi-window operations. In Proc. CHI 97. (pp. 250-257). NY:ACM.

[17] Kaptelinin, V. (2002). UMEA: User-monitoring environment for activities. In Proc. UIST'02 Companion, (pp. 31-32).

[18] MacIntyre, B., Mynatt, E., Voida, S., Hansen, K., Tullio, J., Corso, G. (2001). Support for multitasking and background awareness using interactive peripheral displays. In Proc. UIST 2001, (pp. 41-50).

[19] Malone, T. W. (1983). How do people organize their desks? Implications for the design of office automation systems, ACM Transactions on Office Information Systems 1 (1), 99-112.

[20] Myers, B. (1988). Window interfaces: A taxonomy of window manager user interfaces, IEEE Computer Graphics and Applications, 8 (5), 65-84.

[21] Mynatt, E., Igarashi, T., Edwards, W., and LaMarca, A. (1999). Flatland: new dimensions in office whiteboards. In Proc. CHI'99, (pp. 346-353).

[22] Rekimoto, J. (1999). Time-machine computing: A time-centric approach for the information environment. In Proc UIST'99 (pp. 45-54).

[23] Robertson, G., Czerwinski, M., Larson, K., Robbins, D., Thiel, D., and van Dantzich, M. (1998). Data Mountain: Using spatial memory for document management. In Proc. UIST'98 (pp. 153-162).

[24] Robertson, G. van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D., Gorokhovsky, V. (2000). The task gallery: a 3D window manager. In Proc CHI'00 (pp. 494-501).

[25] Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., Robbins, D., and Andrews, D. (2003). GroupBar: The TaskBar Evolved. In Proc. OZCHI'03.

[26] Teitelman, W. (1986).Ten years of window system – A retrospective view. In Hopgood, F., Duce, D., Fielding, E., Robinson, K., & Williams, A. (Eds.). Methodology of Window Management. Berlin: Springer-Verlag.

[27] XDesk Software (2003), About Virtual Desktop Managers, http://www.virtual-desktop.info.