Scalable Fabric: A Flexible Representation for Task Management

George Robertson, Eric Horvitz, Mary Czerwinski, Dugald Hutchings,
Patrick Baudisch, Brian Meyers, Daniel Robbins, and Greg Smith
Microsoft Research
{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 a flexible visual representation with window management 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 down 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 legacy Microsoft Windows XP TaskBar.

1. Introduction

Twenty years ago, Bannon et al. (1983) observed that information workers often switch between concurrent tasks or activities. In Rooms, Card and Henderson (1987) 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 early work of Bannon et al., numerous virtual desktop managers have been built and each has exhibited some of these properties. Task

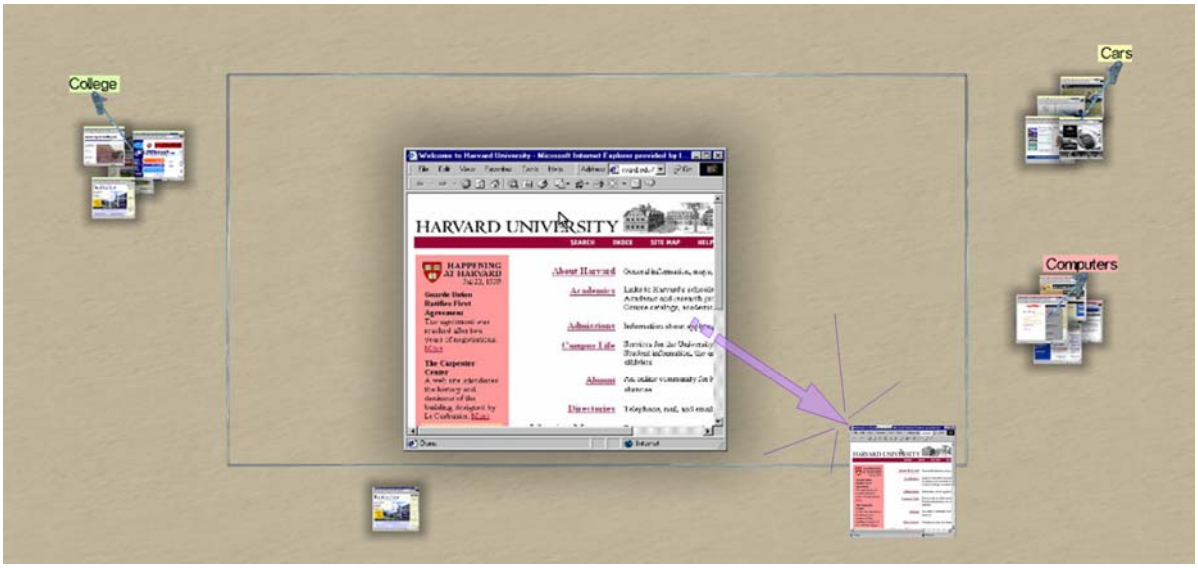

Figure 1. Scalable Fabric showing the representation of three tasks as clusters of windows, and a single window being dragged from the focus area into the periphery.

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 (Czerwinski, Horvitz, and Wilhite, 2004). Czerwinski, Cutrell, and Horvitz (Cutrell, 2001; Czerwinski, 2000; Czerwinski, 2000b) have sought to understand the influence of interruptions on task switching for information workers in order to design user interface 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). Although a larger study is required for verification of these results, 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 (Goldberg, 1983). *Rooms* (Card, 1987; Henderson, 1987) is probably the most well-known of these kinds of systems. A number of virtual desktop managers are currently available, and are described in (XDesk, 2003). 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 (Baudisch, 2001), employing 3D environments as pursued by the TaskGallery (Robertson, 2000) effort, providing a zoomable space as in Pad++ (Bederson, 1994), and the use of time as the main axis and organizing principle (Rekimoto, 1999). Also, tiled window managers (Bly, 1986; Teitelman, 1986) 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 (Bell, 2000; Kandogan, 1997).

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 (Malone, 1983). We have also sought to provide support for task-based visualizations and switching, in a similar vein to the work of Henderson & Card (1987), Kaptelinin (2002), Macintyre et al. (2001) and Robertson et al. (2000).

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 (Czerwinski, 2003).

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 re-

quires forethought to set up. The approach in Scalable Fabric is more dynamic and requires no forethought.

In related work, GroupBar (Smith, 2003) 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 (Robertson, 1998) and Task Gallery (Robertson, 2000) 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 (Czerwinski, 1999). 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 (Mynatt, 1999) and by ZoomScapes' location based scaling mechanism (Guimbretiere, 2001). 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 design 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 Figure 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 will want to 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.

When a window is moved around in the periphery, other windows temporarily move out of the way. This is the same occlusion avoidance behavior employed in the Data Mountain (Robertson, 1998), 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 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).

Natural gestures are also provided to allow users 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 it smoothly integrates user focus of attention 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 (Guimbretiere, 2001). 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, simply 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 could be generated 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. Design and implementation issues

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. We shall now describe the design and implementation issues encountered with each iteration.

4.1. Initial design: image-based prototype

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. This version was implemented in C++ and Direct3D. By building in a 3D environment, we had maximum flexibility in design experimentation. Figure 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.

Figure 2 shows a close-up of a task. Notice that the windows have a colored banner that matches the color of the task marker, and have transparent drop shadows. When the user hovers over a window, a yellow border is drawn around the window and a tooltip shows the full window title. The task marker is a 3D image of a place marker with the name of the task on the tag in the clip. Early testing with the prototype showed that users could not guess what the marker object was.

Figure 2. Close-up of task (initial prototype).

In the first prototype, window occlusion avoidance used the same algorithm that had been used by the Data Mountain (Robertson, 1998). A window being dragged would push other windows out of the way, but they would return to their former position when the moving window went past them. For task group avoidance, we used a simpler algorithm that caused groups to push each other out of the way with no return to former position.

Informal studies were conducted to collect usability issues to drive the second design iteration.

4.2. Second design: live windows

The second design worked with real windows on the Windows Desktop. This version was implemented in C# and the .Net Framework. Because the version was built on top of an existing window manager rather than replacing one, it has some limitations both in behavior and performance. For example, the first prototype preserved window Z-order both in the focus and in the periphery. Using the window manager of Microsoft Windows makes preserving window Z-order extremely difficult, if not impossible, This is an example of a problem that could easily be resolved if Scalable Fabric was built as a replacement for the window manager instead of on top of the window manager.

In the second design, when a window is moved into the periphery, it is replaced with a scaled surrogate window and the "real" window is hidden. This had a side effect of removing windows in the periphery from the TaskBar, which met with mixed reactions from users.

Figure 3 shows a close-up of a task in this implementation. Notice that periphery window text is somewhat hard to read. This implementation did not apply anti-aliasing to the images, which made them hard to read and caused temporal aliasing when windows were dragged. Another problem arose from the method employed in the second design for capturing the content of windows. The window state in this version was captured by copying bitmaps from the desktop image. This means that windows in the periphery were not updated after they had been moved to the periphery. Also, if a window was only partially visible, only the visible part was available, leaving a black region in part of the peripheral window.

Notice that the task marker was redesigned to appear as a flag on a pole. Although this was better received by users, a significant number of users still were not able to identify what the task marker was.

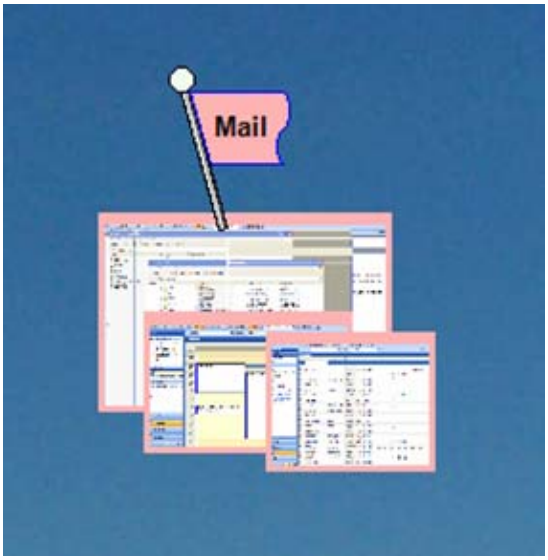

Figure 3. Close-up of task (second design).

Window avoidance behavior in the second design was similar to that of the first prototype, except that windows are of arbitrary sizes and shapes, and were scaled with a preservation of aspect ratios. The algorithm had to be redesigned to ensure that some minimum area of any window remained visible. Task group avoidance remained the same. However, during the first user study, it became clear that having groups push other groups out of the way could invoke problems for some users.

Persistence of window state poses a number of serious challenges. Perhaps the most serious challenge is window-configuration persistence. There is no reliable way to determine how to restore an arbitrary application to the same state in Windows, because internal application state (*e.g.*, which files are open) is not accessible. Rather than solve this problem, the second version of Scalable Fabric only saves window position, size, and title. When Scalable Fabric is restarted, if an open window of the same title is discovered, it will be restored to the last state it was rendered within Scalable Fabric. If the window is not present, Scalable Fabric does not try to start an application and restore its running state.

The second version of Scalable Fabric also allowed for restoring and changing state for multiple monitors and changing display resolutions. Scalable Fabric does this by preserving window size and position information as percentages of display size. When display resolution changes, windows and tasks are moved to the appropriate location.

In Section 5, we shall present the results of a comparative user study of the second version of Scalable Fabric. The study revealed several usability issues which drove the third design. Before moving on to user studies of versions of Scalable Fabric, we will review the third version of the system.

4.3. Third design

The third version of Scalable Fabric was developed as a set of refinements on the second design. We solved several of the most obvious problems by using the Windows PrintWindow function instead of copying bitmaps from the desktop image. Although this process is somewhat slower, it provides a full image regardless of how much of the window had been visible. Also, "real" windows are no longer hidden, but rather moved off-screen. This solved a number of problems with applications that made assumptions about hidden windows. For example, we found that Internet Explorer sometimes destroyed hidden windows assuming they were from pop-up ads. But no longer hiding the windows meant that they remained on the TaskBar when in the periphery – and some users did not like this change. In the third version, windows in the peripheral tasks are updated periodically.

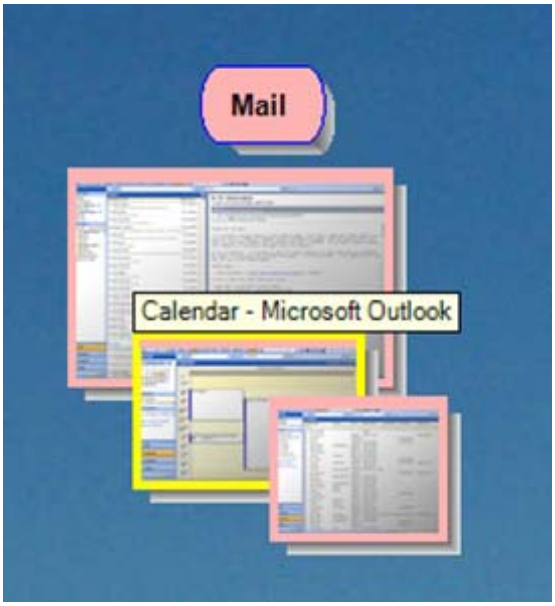

Figure 4. Close-up of task (third design).

Figure 4 shows a close-up of the redesigned appearance of windows and task markers, with the cursor hovering over one window to show its title tooltip. Windows are now anti-aliased for more readability, which also prevents them from flickering while being moved. A shaded gradient was added over the window and drop-shadows were added under the window to make the windows more distinguishable. The task marker was simplified to a two-stage marker. Most of the time the task marker appears as displayed in Figure 4. However, if the user hovers over the marker or moves a window into the task group, a box appears as rendered in Figure 5.

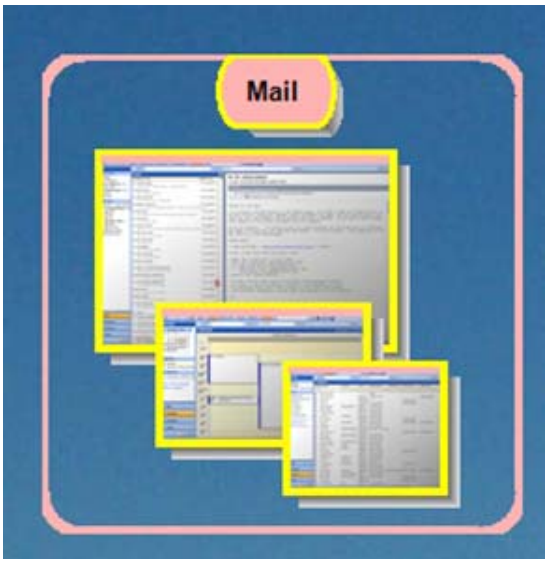

Figure 5. Task highlighting during hover.

Window occlusion avoidance was not changed in the third version of Scalable Fabric. However, we modified task group avoidance to address a problem identified by testing. In the new design, the moved task automatically avoids tasks that it encounters rather than pushing them out of the way (*i.e.*, it moves around them). Users appear to prefer this approach.

ALT-TAB ALTERNATIVE

The Windows OS provides users with keyboard commands that enable them to switch among windows. We found that a number of users desired some analog to these commands to allow for keyboard access to tasks and windows within tasks. In Windows, Alt-Tab is the standard keyboard shortcut for switching between windows. It displays a list of windows which the user can sequence through by repeatedly hitting Tab, and then releasing Alt to active the selected window. Until the third iteration, Scalable Fabric had not included keyboard navigation facilities, limiting users to interacting with peripheral windows via mouse interaction.

We implemented three keyboard navigation mechanisms to explore this design space and then conducted informal testing on them. In the following, we describe the most successful of these three alternatives, which we integrated into the third iteration of Scalable Fabric.

To start a selection, the user holds the `Win` key and taps any arrow. The peripheral object furthest in the selected direction (for example, if the left arrow key is pressed, the leftmost object) is selected. From there, the arrow keys move to the closest object in that direction. We define *closest* as "the object located less than 45 degrees from the selected direction which also has the shortest Euclidean distance" (see Figure 6). Once inside a group, the user can hit enter to select the entire group, or use the arrow keys (without the windows key depressed) to select through individual windows. An additional feature is switching to the previous selected group by pressing `Win`+`G`, which is akin to `Alt-Tab` for focal windows as one tap of this combination results in selecting the previously activated window. In addition Alt-Tab allows users to cycle through focal windows.

The advantage of this style of navigation is that we allow users to use their spatial memory to access objects. The potential disadvantage is that more keystrokes may be needed to access an object than would be required with the standard `Alt-Tab` mechanism.

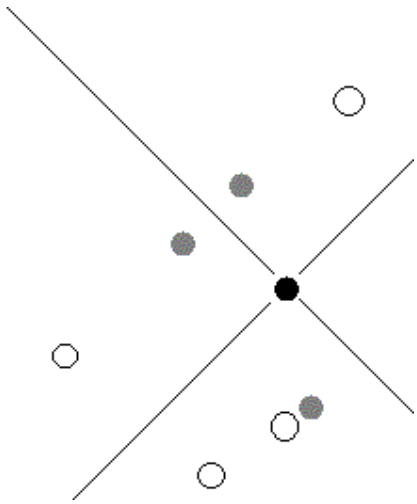

Figure 6: The black dot is the currently selected object. The gray dots represent the next object to be selected when the corresponding arrow key is pressed, and the white dots represent other objects. The lines through the black dot represent the 45 degree lines.

4.4. User response

Approximately 150 people have used Scalable Fabric over the last 8 months. 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 it 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 in order to get better control.

5. Initial laboratory user study

We now turn to user studies of Scalable Fabric. We shall start with results from a laboratory investigation of the second iteration of Scalable Fabric. We undertook this study to collect basic feedback on the overall concept and to seek guidance on future design refinements. In the laboratory investigation, we created 3 tasks consisting of between 2 and 3 documents each, matching what we saw on average from our informal observations from an earlier diary study (Czerwinski, Horvitz, & Wilhite, 2004). The tasks consisted of a "Spreadsheet", a "Joke" and an "Image" task. The Spreadsheet task required participants to go to selected cells in an Excel spreadsheet as indicated by a Word document, copy the contents of the cell at that location (a 9 digit random number), and paste it both in the Word document and in another Excel spreadsheet. The Joke task required users to identify typographical errors in a list of jokes in a Word document, copy them and paste them and the page number on which they occurred in an Excel spreadsheet. The Image task required participants to modify images in PowerPoint and in Paint based on instructions in a Word document. Two isomorphic sets of each of these tasks were devised so that they were of approximately equal difficulty (e.g., the random numbers in the Excel spreadsheet were rearranged, as were the selected cells, both Joke documents consisted of 2300 characters and had 23 typographical errors but were different jokes, and the instructions for how to modify each image simply asked the user to use different simple shapes or colors).

5.1. Method

Eighteen participants (half female), all multiple monitor users and very experienced MS Office users as identified by a validated screener, were recruited for this study. Participants were 34 years old, on average, had used the computer for an average of 14 years, and said they task switched among 6 tasks daily, on average.

Participants were given instructions about the overall study procedure and then allowed to read a brief overview of how each tool, the TaskBar or Scalable Fabric, worked before proceeding. All participants were very familiar with the TaskBar and knew of most of its features, even if they didn't choose to use them. In order to ensure that they learned how to use Scalable Fabric, users were guided through the grouping and layout of the documents that formed each of the three tasks for the study. In the TaskBar, participants could arrange the items in the TaskBar by application (as supported in the software) but not by task. However, we did allow them to lay out and size their task windows in a way that was best suited to each task before beginning. The TaskBar was laid out vertically along the left-most bezel of the left-most monitor (a triple-monitor display setup was used for the study). While running Scalable Fabric, the TaskBar was laid out horizontally and put on "auto-hide," so that users never saw it and they were instructed to solely interact with Scalable Fabric. Finally, the "agglomeration by application" mechanism in the TaskBar was turned on so that window tiles of like applications were juxtaposed, but they did not collapse into a single tile menu of all application windows. We did this to the benefit of the TaskBar, as previous studies (Czerwinski, 2003) had shown that this feature impedes performance when a large number of windows are left open.

In order to ensure that participants had to task switch between tasks, the experimenter interrupted them at set places in their tasks. Five task switches were required in order to carry out all three tasks to completion, the first three of these were guided by experimenter interruptions between the Spreadsheet task and the Joke task when participants were approximately ¼ of the way through each, and then again at approximately ¾ of the way through the first task (Spreadsheet). After the first 3 interruptions, participants were told to finish the Joke task that they were in, go back and finish the Spreadsheet task, and then switch to the Image task and carry it through to completion. A twenty minute deadline procedure was used for each of the three tasks to keep the session length under two hours. All participants completed their tasks before the deadline.

The study was run on two identical, late model Compaq Evo machines with triple flat panel LCD monitors running at 3840 x 1024 resolution. Late model MS keyboards and the IntelliMouse were used for input. Windows and Office XP™ comprised the base OS and applications used in the study. The order of software tool used and task set were counterbalanced across participants. Participants were run in pairs each session.

Dependent measures collected included task time, subjective satisfaction responses to a questionnaire presented after using each tool, and overall tool preference. Task times were recorded using a countdown program on the participants' machines. A log of users' activities in terms of window management and group interaction was collected; analysis of that data is ongoing.

5.2. Results

A t-test of the task times revealed no significant task advantage for Scalable Fabric, $t(17)=-.03$, $p=0.5$, one-tailed. Scalable Fabric's average task time was 13.28 minutes, while the average for the TaskBar was 13.25. The task time data including one standard error of the mean in each direction are presented in Figure 7.

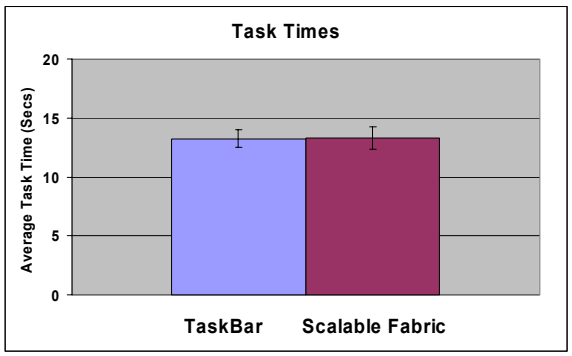

Figure 7: Average task times +/- one standard error for TaskBar and Scalable Fabric.

Survey Question (1=Disagree, 5=Agree)	TaskBar	Scalable Fabric
Task switching was easy to perform using the…	2.95	4.26
It was hard to go back and forth between my various windows and applications using…..	3.32	1.84
I was satisfied with the functionality of the ….	2.68	3.78
The TaskBar/Scalable Fabric is an attractive innovation for Windows.	3.16	4.47

Table 1: Average satisfaction ratings for the TaskBar and Scalable Fabric. All ratings were significantly in favor of Scalable Fabric at the p<.05 level.

With regard to the overall satisfaction with the software, participants preferred Scalable Fabric over the TaskBar. Paired t-tests were used to analyze the ratings and all differences were significant ($p<.05$), using the Bonferroni correction for multiple tests.

Despite these positive ratings, there were several usability issues identified with this initial version of Scalable Fabric. First, because of its novelty, some users still minimized single windows to switch entire tasks (multi-window by definition) even after they were instructed to use the markers. Hence, the markers themselves needed better "grab-ability" and visibility. Users mentioned that they sometimes did not understand what the markers were meant to represent, as well. In addition, the dragging of windows into the periphery was not as smooth as users would have liked, and sometimes the window scaling into the periphery did not work immediately, causing users to hesitate or try again. Finally, most of our participants wanted an Alt-Tab--style navigation mechanism to work in Scalable Fabric, both for moving between groups and also within a group.

As for the TaskBar, the users typically resized it to be very wide so that they could read more of the titles, but still often complained that the "tile" naming on the TaskBar was inconsistent. For example, some Excel spreadsheet tiles had the name of the spreadsheet as the text label with a small Excel icon next to it, while others had a large Excel icon and simply read "Microsoft Excel". Users did not understand this distinction.

5.3. Discussion

Although the lab study suggested to us that Scalable Fabric was easily learned and considered valuable by the participants, usability issues were apparent and needed to be addressed. Without this redesign, it is unlikely that we will observe performance improvements over the familiar TaskBar mechanism. Still, we were encouraged that the novel metaphor of Scalable Fabric was not significantly slower than the TaskBar in terms of average task performance. Users commented that the tasks and interruptions forcing the switches were similar to what they experienced in the real world, so we feel we succeeded in simulating an information worker's daily task juggling at an abstract level. The study provides initial evidence that software tools like Scalable Fabric can provide user assistance as users manage multiple, complex tasks.

After making the required design changes (see section 4.3), we ran a second study evaluating Scalable Fabric, but this time we chose to do the examination *in situ*, over a 3 week period, with users' real tasks.

6. Longitudinal field study

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 performed a longitudinal field study on a small number of subjects over a reasonably long time period.

6.1. Method

Thirteen participants, aged between 20 and 60 and all male, were recruited to participate in the study. The participants volunteered from an internal alias of individuals interested in learning more about Scalable Fabric. Some of these participants had seen the first version of Scalable Fabric, but none of them had used the system since the redesigns had taken place. All participants were technologically savvy, so this sample should be considered with caution when considering how to generalize our initial findings from the study.

A field study methodology was utilized in order to examine the usefulness and usability of Scalable Fabric compared to the existing TaskBar. We used an *in situ*

method to study the use of Scalable Fabric in order to establish how important the new Scalable Fabric features were *using the participants' own work information and habits*. If, after approximately one to three weeks of use, participants were satisfied using the task grouping features of Scalable Fabric, this would provide evidence of the system's usefulness.

The participants were emailed an introductory email, alerting them to the availability of the new version of Scalable Fabric for download, and that we would like them to participate in a survey at the end of trying to use the system for 3 weeks. After the 3 week time period, we issued the survey.

6.2. Results

Once it was installed, all users were able to easily learn how to use the grouping features in Scalable Fabric. We only received one email after sending out the installation instructions and FAQ, and this was a user who did not understand he needed to drag windows into groups in the periphery (*i.e.*, had not read the instructions). In addition, users were able to easily integrate Scalable Fabric into their existing work practices, as evidenced by their comments and grouping habits. This is what we had hoped to observe, and it was our main design goal.

After using Scalable Fabric for about three weeks, thirteen volunteers completed a user satisfaction survey and entered details about how they used the system. All but one of the respondents had used Scalable Fabric for longer than one day. Unfortunately, the responses were anonymous and it was impossible to track the lone user who had not used the system very long and remove his data. However, all of the users who reported using the system for some length of time did so for more than two days, as can be seen in Figure 8, and 3 participants were still using the system after 3 weeks. Responses to the survey questions are shown in Figures 8 through 12.

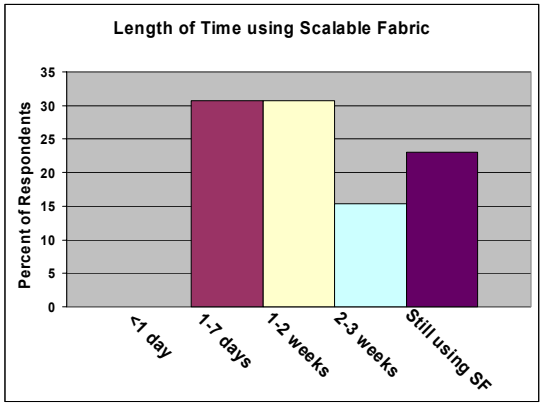

Figure 8. Length of time using Scalable Fabric.

Participants indicated that the initial assignment of windows to tasks in the periphery was somewhat difficult and time consuming. On a scale of 1=very difficult to 5=very easy, participants rated this initial organizational task an average of 2.07. The distribution of ratings can be seen in Figure 9. In addition, we queried users about the length of time it took to set up their systems, and whether or not they agreed that the amount of time spent was appropriate. On average, users leaned slightly towards not approving of this time expenditure, average rating=2.4 (1=strongly disagree, 5=strongly agree). These findings are shown in Figure 10. In fact, when asked if Scalable Fabric made managing windows and task switching easier, participants disagreed slightly, with a rating of 2.8, on average. Perhaps this was because all of the users still felt the need to utilize the TaskBar at least some of the time (5 continued to use the TaskBar, while 7 reported using it sometimes). The goal of Scalable Fabric was for the task clusters themselves to be used for task switching. If users were still needing to use the TaskBar for the Start menu, or the system tray, etc., this may have diminished the value of having the tasks laid out in the periphery.

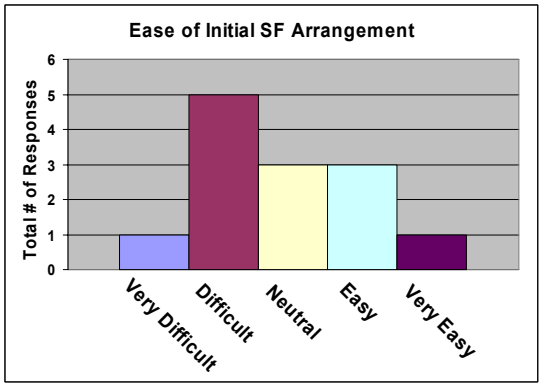

Figure 9. Ease of initial arrangement.

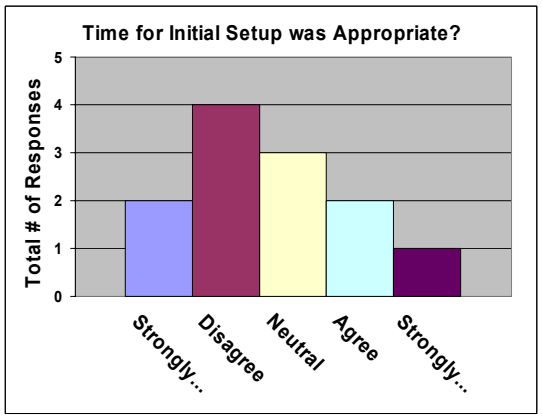

Figure 10. Time for initial setup was appropriate – from "strongly disagree" to "strongly agree".

Half of the users reported an average of 1-2 windows per task, and half reported 3-5 windows per task, as can

be shown in Figure 11. The median response was for 1 or 2 windows to comprise a task in the periphery. In addition, users reported moving windows between tasks infrequently (6 users reported never moving windows between tasks and 5 reported they sometimes did this). Finally, 8 of the users kept the boundaries of the focus area visible, while 4 users reported hiding it.

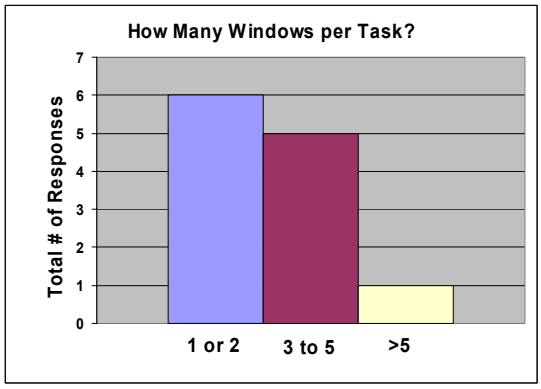

Figure 11. How many windows per task?

Users provided us with some comments about the Scalable Fabric concept at this stage in its iterative design. One participant stated that he thought the concept was great, but cited technical glitches with windows disappearing off the screen (this is a known bug with Visual Studio) and with variable sizing in the periphery. Another user remarked that he appreciated keeping windows scaled down and visible in the periphery instead of minimized in the TaskBar, but that he didn't see much need for grouping them into related task items. Three users said they'd have continued using Scalable Fabric, but the performance was still too slow. One participant mentioned preferring a central, unified place for storage as opposed to the desktop periphery.

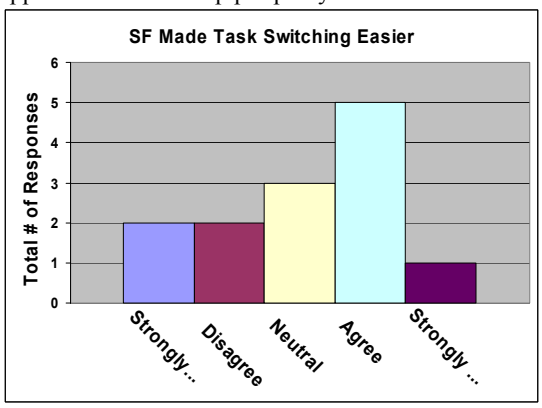

Figure 12. SF made task switching easier – from "strongly disagree" to "strongly agree".

In all, we received a mixed review of the third design iteration of Scalable Fabric. From the field study, it is clear that we will need to enhance system performance for windows updating in the periphery, track down the Visual Studio bug and possibly consider alternative configurations of the periphery boundaries so that less screen real estate need be devoted to them. Currently users can delegate one side of their display to the periphery, but the system does not ship that way by default and it may be that users do not discover that they can resize the boundaries to suit their preferences.

6.3. Discussion

The lab study provided initial evidence that some of Scalable Fabric design decisions were deemed valuable by users. For a first iteration design, users were able to easily figure out how to use Scalable Fabric and were using the grouping features for their common tasks. Given the lack of ethnographic data available to designers of task management systems, we conducted a second, longitudinal study with 13 participants using their real systems and tasks. While we did confirm that we had solved many of the usability issues identified in the first study, this second study revealed new areas for continued design iteration. Specifically, 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 our future work.

We also learned that users find initial configuration taxing. This finding has stimulated us to pursue machinery for providing new tools for clustering and configuration of tasks (not yet implemented), including approaches that provide for regularized handling of specific forms of tasks. For example, there is opportunity to provide defaults, as well as a means for encoding a user's preferences about prototypical tasks. Let us consider the case of providing a canonical configuration for a "messaging and scheduling" task, *e.g.*, capturing the services provided by the Microsoft Outlook application. Scalable Fabric could include a standard geometry for such a task as a main "anchor" window, representing an email inbox view, and a cascade of open email messages, aligned adjacent to the main anchor. Such methods might ease the difficulty of setting up windows and tasks.

7. 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 have shown 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.

8. References

Baerbak Christensen, H. (1997). *A software development environment based on a geographic space metaphor.* Technical report, Univ. of Arhus.

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.

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

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

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

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

Bury, K. F., Davies, S. E., and Darnell, M. J. (1985). Window management: A review of issues and some results from user testing. *IBM Human Factors Center Report HFC-53,* San Jose, CA.

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.

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.

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.

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.

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.

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

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

Goldberg, A. (1983). *Smalltalk-80*. NY: Addison-Wesley.

Grudin, J. (2001). Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *Proc. CHI'01*, (pp. 458-465).

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.

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.

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

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

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).

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.

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

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

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

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).

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).

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

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.

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