Clipping Lists and Change Borders: Improving Multitasking Efficiency with Peripheral Information Design

Tara Matthews **Mary Czerwinski** **George Robertson** **Desney Tan**
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
tmatthew@cs.berkeley.edu,{marycz, ggr, desney}@microsoft.com

ABSTRACT
Information workers often have to balance many tasks and interruptions. In this work, we explore peripheral display techniques that improve multitasking efficiency by helping users maintain task flow, know when to resume tasks, and more easily reacquire tasks. Specifically, we compare two types of abstraction that provide different task information: semantic content extraction, which displays only the most relevant content in a window, and change detection, which signals when a change has occurred in a window (all designed as modifications to Scalable Fabric [17]). Results from our user study suggest that semantic content extraction improves multitasking performance more so than either change detection or our base case of scaling. Results also show that semantic content extraction provides significant benefits to task flow, resumption timing, and reacquisition. We discuss the implication of these findings on the design of peripheral interfaces that support multitasking.

Author Keywords: Information visualization, peripheral displays, abstraction, multitasking

ACM Classification Keywords: H5.2 *User Interfaces*– Graphical user interfaces. H5.m *Miscellaneous*.

INTRODUCTION
Information workers often balance many interruptions and tasks. In fact, a recent study found that information workers kept an average of 10 "working spheres," or basic units of work, active at once [4]. This is not surprising given the many studies on task switching showing that users spend as little as 3 minutes (on average) on each task before switching, and get interrupted at least once per task [2,4,13].

CHI 2006, April 22–28, 2006, Montréal, Québec, Canada.

These studies point to several problems multitaskers continue to battle due to inadequate software support. First, interruptions plague longer-term tasks undermining user concentration and task progress, because users are often unable to determine which interruptions need to be handled immediately [2,18]. This makes it difficult for users to *maintain current task flow*. Second, successful task completion requires knowing when to step out of the current task and return to a paused task, which we call *resumption timing*. For example, Czerwinski *et al.* [2] found that people often set aside tasks while waiting for some external event (*e.g.*, an email to arrive from a co-worker), but wanted to resume as soon as the event occurred. Third, people have trouble getting back on task after shifting their attention away (*i.e.,* it is difficult to *reacquire tasks*) [2].

We believe that providing relevant task information in a glanceable, low-attention manner is critical to solving these multitasking problems. However, little is known about *what* information is needed and how to provide it to best improve a user's ability to maintain current task flow, manage task resumption, and reacquire tasks in multitasking situations. In this paper, we explore abstraction in peripheral interfaces, studying the performance effects when these interfaces convey varying types of task information.

Figure 1: Clipping Lists interface showing our study's Quiz task with a progress bar indicating the next quiz module is loading, and Web page text to help users recognize them.

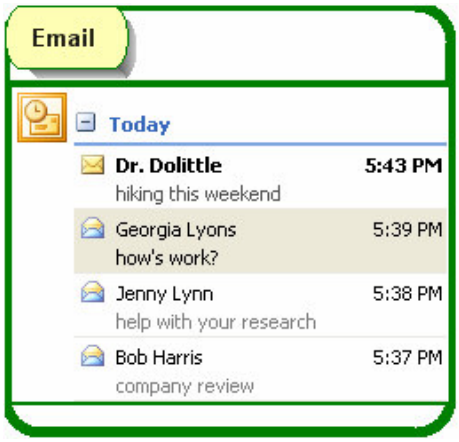

Figure 2: Clipping Lists with Borders shows an email inbox and indicates with a green border that new email arrived.

Peripheral displays convey information in a low-attention way outside of a focal task. They often employ *abstraction* to reduce the fidelity of raw information so that is easier to read at a glance. Abstraction is a design mechanism for reducing the amount of information shown and/or transforming the information to a different form. Although several researchers have argued that abstraction improves glanceability and enables people to more quickly understand information [15,16], existing research has not established strong guidelines for *what types* of abstraction lead to easier understanding without losing too much information. In our work we explore this issue by examining two types of abstraction, semantic content extraction and change detection, in multitasking situations.

Semantic content extraction refers to selecting and displaying only pieces of information from an original window that are relevant to task flow, resumption timing, and task reacquisition. In our interfaces, we show in the periphery lists of rectangular window "clippings," which we call *Clipping Lists* (see Figure 1). Change detection provides one extra bit of information to the user, signaling when a change has occurred within a window. In our interfaces, we use *Change Borders*, colored highlights that appear around peripheral windows, to indicate changed content (see Figure 2). We conducted a lab study comparing four interfaces that introduced these abstraction types to the Scalable Fabric task management system [17]. In performing these comparisons, we were not interested in which form of abstraction shows *more* information, but rather which shows the most *relevant task* information in a glanceable way.

The main contribution of this paper is an empirical demonstration that semantic content extraction is more effective than both change detection and our baseline case of scaling peripheral windows in improving multitasking efficiency. We show that since our implementation of semantic content extraction significantly reduces task time, task switches, and window switches, it benefits task flow, resumption timing, and reacquisition. We use these findings to discuss how to design peripheral displays that aid multitasking and task flow, so that users focus their cognitive resources on the task at hand, instead of on task management.

RELATED WORK

Central to the notion of multitasking is a concept known in psychology as prospective memory [3], or remembering to remember. Successful prospective memory requires effective task resumption timing, or recalling tasks at appropriate times, and can help dramatically in reacquiring tasks.

A number of studies have shown prospective memory failures to be a major issue for information workers who multitask [1,10,14,19,21]. In fact, other studies have examined how users remind themselves about tasks [6,9], such as creating Web pages with "to do" lists and emailing reminders. Clearly, people spend a great deal of time devising solutions to multitasking problems. These studies indicate that users need better software support for prospective memory when doing multiple tasks, which we study in our designs.

We assert that one of the reasons software support is inadequate for multitasking is that it is isolated to a single application (*e.g.,* the "to do" list in a software calendar). However, tasks span multiple applications. In a diary study of information workers, Czerwinski *et al.* [2] found that users wanted an overarching application to keep track of tasks across applications. We address this issue by using Scalable Fabric [17], a system for organizing task windows across applications, as a basis for our designs.

Several other projects have proposed systems that allow users to manage tasks by interacting with information across applications. Hutchings and Stasko [8] and Tan *et al.* [20] presented techniques for selecting a rectangular portion of a window so that only the window's relevant information is visible (a form of semantic content extraction). Hutchings and Stasko's shrinking windows operation replaced an existing window with the user-selected portion. Tan *et al.*'s WinCuts created a new window with the user-selected portion, leaving the original window for the user to use or minimize. These two projects allowed users to extract semantic content, much like our Clipping Lists interface. We extend this work by studying the effects of semantic content extraction on multitasking efficiency in conjunction with change detection.

Because we believe that users will benefit the most from information that does not divert attention from the focal task, we intentionally avoid distracting notifications and explore our designs within peripheral displays. Peripheral display design has been explored in many projects (see [12] for a survey). For example, Kimura [11] uses a peripheral display to help users keep track of multiple tasks. In their system, a task is represented as a montage of images (*e.g.,* application windows and notification icons) gathered from past activities. Montages are displayed on a large, digital white board on which users can interact. Much like our designs, Kimura provides constant, visual reminders about various tasks. One improvement in our system is that our

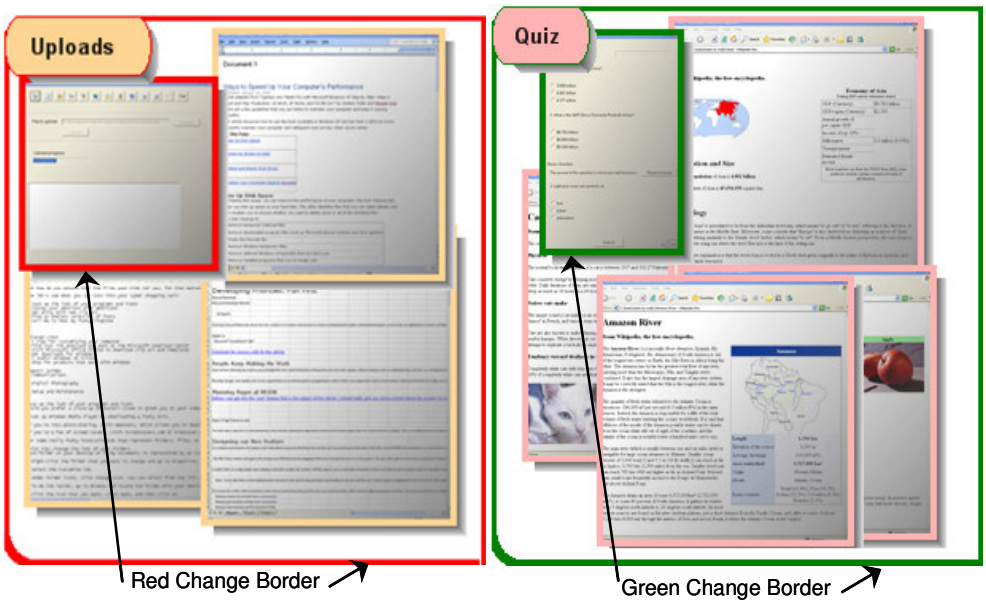

Figure 3: Scalable Fabric with Change Borders: (left) showing upload in progress; (right) indicating a quiz module has finished loading. Peach and pink window borders indicate to which task the window belongs.

peripheral windows are live and updating. This provides an environment for monitoring new task information. We extend this work by empirically comparing live window displays that abstract information differently.

Pederson's AROMA project [15] distinguished three types of abstraction in peripheral displays: degradation (*e.g.,* pixelation, thresholding), feature extraction (*i.e.,* singling-out certain parts of an information source for display), and radical abstraction (*i.e.,* extracting features from an information source and displaying it in a new, symbolic form). AROMA specifically explored the use of "radical abstraction" to support remote awareness. AROMA combined audio and video streams into a single "bustle factor" bit aimed at providing users with a general sense of how much activity was happening in another location. Since no formal evaluation was reported, we cannot conclude how useful radical abstraction was for AROMA's design goals.

Plaue *et al.* [16] compared three forms of electronic information to see how quickly each conveyed information: text-based, Web portal, and pictorial displays. The pictorial display, InfoCanvas, is a peripheral display that uses pictures to represent multiple streams of information, such as weather, traffic, and stock prices. After staring at each display for 8 seconds, users were tested for their immediate recall of the focally displayed information. Results showed that the pictorial display enabled users to recall significantly more information. In contrast, we compare the effectiveness of our abstracted interfaces to convey information *peripherally*, in a more realistic usage scenario.

FOUR INTERFACE DESIGNS

Here we present four interfaces that instantiate our ideas around semantic content extraction and change detection. First we describe Scalable Fabric, a task management system which we used as our baseline and augmented to include different types of abstraction. Next, we describe Clipping Lists, which replaces the scaling techniques used in Scalable Fabric with semantic content extraction. Finally, we present Change Borders, a change detection technique, which we apply to both Scalable Fabric and Clipping Lists. For each interface, we explicitly designed updates to be subtle, as we are interested in comparing the abstraction techniques rather than how well each interface attracts user attention (*e.g.*, through distracting notifications).

Scalable Fabric

Scalable Fabric [17] provides task management support by spatially organizing shrunken versions of windows into tasks in the periphery and enabling window management on a task level. Users interact with windows in a central focus area of the screen in a normal manner, but when a user moves a window into the periphery, it shrinks. The window "minimize" action is redefined to return the window to the user-chosen location in the periphery, rather than hiding the window. Placing a set of windows near each other in the periphery dynamically creates a task composed of them. Clicking on a minimized task's name restores all its windows into the central focus area. This is the equivalent of switching rooms in the Rooms system [7]. Clicking again on the focal task's name minimizes all its windows to the periphery. Windows from all tasks are visible at all times (either in the focus area or as shrunken, peripheral windows), enabling easier task switching.

Scalable Fabric provided the baseline comparison within our study. In a previous study, Scalable Fabric was found to perform as well as Windows [17] and was qualitatively preferred. Comparing our three new interfaces to Windows would not have been a fair comparison, as they introduce task management support in addition to abstraction. By comparing them to Scalable Fabric, which uses a simple abstraction technique (*i.e.,* scaling or shrunken windows), we gained additional insights about abstraction.

We compare scaling as a baseline to semantic content extraction and change detection. A shrunken peripheral window enables users to see the general layout of the window, major color schemes, and graphics, but virtually none of the content is legible. The benefits of shrunken windows are that they convey the window's spatial layout, they may

enable easy recognition of windows for reacquiring tasks, and large updates are visible.

Clipping Lists

Our Clipping Lists interface uses semantic content extraction, replacing the shrunken peripheral windows of Scalable Fabric with vertical lists of window "Clippings." Here, semantic content extraction refers to selecting and displaying Clippings, rectangular sub-regions of relevant information from the original window. For example, Figure 1 shows Clippings of a progress bar and Web page text for a Quiz task. This enables users to monitor the progress bar so they know when they can return, and to easily identify Web pages by their title text. We also place an application icon next to Clippings to further aid identification (*e.g.,* Figure 1 shows Internet Explorer icons next to Web page Clippings).

To create a custom Clipping, the user hits a modifier sequence ('Windows-Key + Z') at which time a red rectangle appears. The user drags or resizes the rectangle over their desired Clipping. To capture the Clipping, the user hits the modifier sequence again. From then on, the Clipping will appear in the periphery whenever the window is minimized. This interaction was inspired by the WinCuts system [20].

In our system, users manually create Clippings because it is not currently technically feasible to automatically determine what window content is most relevant to a user. However, we believe future research might enable useful and meaningful automation.

When users do not manually create a Clipping, a 200 x 32 pixel Clipping is automatically made of the top-left of a window, usually including a file and application name. We chose this default area since it most consistently identifies the window contents.

Semantic content extraction provides a more readable form for peripheral task information, which may help users determine when to resume a task and recognize windows for easier task reacquisition. However, it may also increase the cognitive overhead of monitoring the periphery by providing too much information. In addition, it removes spatial layout information about a given window, the relative importance of which we examine in our study.

Change Borders

We chose to visually represent changes with Change Borders, colored borders that appear around peripheral windows or Clippings when the system detects that the window content has changed. The borders are red while changes are happening (*i.e.,* window pixels have changed recently) and green when changes are complete (*i.e.,* no window pixels have changed for a certain period of time). We added Change Borders to both Scalable Fabric and Clipping Lists, as described below.

Scalable Fabric + Change Borders

Our third interface adds change detection to Scalable Fabric using Change Borders. To illustrate, Figure 3a shows a red Change Border around an upload tool, indicating a document upload is in progress. Figure 3b shows a green Change Border around a quiz tool, indicating that a quiz module has finished loading. Here, Change Borders may help users know when the upload or quiz loading is done and they can resume either task.

Clipping Lists + Change Borders

Our fourth interface combines change detection and semantic content extraction, using both Change Borders and Clipping Lists. Figure 2 shows an email has arrived: the Change Border allows the user to easily glance and see that an email has arrived and the Clipping allows the user to determine if it is important to read.

In general, knowing when a change has occurred within a particular window can be useful in helping users know when to resume a paused task. There are many situations in which knowledge of changes would be useful: when waiting for email, documents to upload, files to be checked in to a version control system, a compiler build to complete, a large file or Web page to finish loading, *etc.* However, Web pages and other windows might change because of ads, which may render change detection less useful. Given that only one bit of information is conveyed (*i.e.,* that a change has occurred), we assert that the cognitive overhead caused by change detection interfaces is low. In our study, we examined whether change detection provides enough information to also improve multitasking performance, or whether it would be considered annoying.

IMPLEMENTATION

All interfaces were implemented on top of our Scalable Fabric implementation [17]. For Change Borders, we detected changes with a pixel-based image difference of two window images, taken 1 second apart. The images were sampled (*e.g.,* every 10^{th} pixel), and if any of the sampled pixels changed, the window was flagged as having a change in progress. This caused a red border to appear around the window. Five seconds after these changes stopped, the window was flagged as having completed changes, causing the border to turn green.

Though the image differencing method could potentially be fairly accurate, we wanted to test this with 100% accuracy in our lab study. Therefore, we implemented message passing between task windows and Scalable Fabric. Whenever a change occurred in the study tasks, the task window (which we controlled) sent a message to Scalable Fabric indicating a change was in progress or completed. We used messages rather than image differencing during the study.

For Clipping Lists, we made two changes to the Scalable Fabric baseline: (1) we enabled user selections of a window region and then rendered only that portion of the window in the periphery; and (2) rather than allowing user arrangement of peripheral windows, we automatically arranged Clippings to be left justified and stacked vertically (users could still change the vertical position of a Clipping). This

modification was important because we could not allow windows to overlap, as every part of a selected Clipping is considered relevant to users.

USER STUDY

We conducted a user study comparing our four interfaces, Scalable Fabric, Clipping Lists, Scalable Fabric with Change Borders, and Clipping Lists with Change Borders, in a simulated multitasking situation. The study tasks simulated real office knowledge work (*e.g.,* monitoring email, uploading files, filling out web forms, and arranging images to make a nice graphic layout). We hypothesized that Clipping Lists would be more useful than Scalable Fabric in multitasking work because the interface extracted the semantic information that would allow users to stay in their current task flow longer, while also knowing the optimal time to switch to a high priority task. We also hypothesized that change detection, as we designed it, would be a lightweight, glanceable method that would further increase multitasking effectiveness while preserving current task flow.

Participants

We recruited 26 users (10 female) from the greater Seattle area who had moderate to high experience using computers and were intermediate to expert users of Microsoft Office-style applications, as indicated through a well validated screener. Users ranged in age from 23 to 53, with an average of 38. They had used a computer for an average of 18 years. Users received software gratuities for their time.

Equipment

We ran the study on a 2.8 GHz Pentium 4 Compaq PC with 2G of RAM and two NEC MultiSync LCDs 1880SX set at 1024 x 768 display resolution. Users provided input with a Compaq keyboard and Microsoft Intellisense mouse.

Tasks

To simulate multitasking, users performed three tasks and also read email. Two tasks were higher priority (Quiz and Upload) and involved waiting for updates. We instructed users to work on the third, lower priority task (a Puzzle) while waiting for high priority updates, to monitor the Quiz and Upload tasks in the periphery, and to return to them as soon as updates were complete. Email provided necessary documents and information for priority tasks, and distracter email, in order to mimic the real world and test how well our interfaces enabled users to ignore irrelevant information.

We told users to complete all three tasks as quickly as possible and that they were being timed. Study timing software was run on the top of the left monitor. Users stopped timers for each task by clicking on labeled "task done" buttons.

To attain high external validity, it was important for the tasks to mimic real world tasks, including multitasking, tasks of varying importance, and interruptions. Also, the tasks needed to be engaging enough that users would not mind repeating slight variations of them for each interface. After the study, users indicated that we had successfully captured the sorts of task and email juggling they do in a typical work day. Though a lab study necessarily reduces realism by simplifying work contexts, our study succeeded in drawing out important issues that peripheral display designers can use. Next, we describe the details of each task.

Quiz

The Quiz task was labeled a high priority task, and it involved waiting for updates. The task was composed of 5 windows: a Quiz Tool and 4 Web pages containing graphics (all of which were open and pre-arranged).

Users used our Quiz Tool to answer Web-based, Wikipedia-derived questions, as might be indicative of a typical research project. Answers were not known by any of our participants (*e.g.,* what is the length of the Amazon River). A quiz included 4 modules, each with a different topic (*e.g.,* Cats, Asia, Apples, and the Amazon River). Each module had 2 questions about its topic. We provided the answers in the Web page titled with the topic name. Within the Web page, answers were bolded and easy to find so that locating them was neither cognitively nor mechanically challenging.

Each module also had a bonus question about a different topic not answered in any of the provided Web pages. Users clicked a button to request the answer via email from their co-worker Jane Morgan (which arrived between 12 and 90 seconds later). When all three module questions were answered, users clicked the "Submit" button. After the first three modules, this caused the next module to load, during which time a progress bar indicated the loading time (between 30 and 120 seconds). While waiting for the next module to load or for an email, users could work on other tasks, but we instructed them to return to the Quiz as soon as possible (*e.g.,* when the bonus answer had arrived in email from Jane Morgan).

After the last module was complete, clicking "Submit" displayed the message, "Task complete. Stop task timer." Users would then stop the Quiz task timer.

Uploads

Like the Quiz, the Upload task was labeled high priority, and it involved waiting time. The task included 4 windows: an Upload Tool and 3 text-based documents (1 Word, 1 Excel, 1 Notepad).

Users used our Upload Tool to upload 5 documents in a pre-specified order. The Upload Tool allowed users to use a browse dialog box to find documents, a progress bar to monitor the upload, and a text box listing already uploaded documents. While uploading, the progress bar indicated the waiting time (30 - 120 seconds). Users started the task with 3 of the documents. The other 2 were missing and would arrive via email from a co-worker, Jane Morgan. The 3 open documents had the upload order specified in their first line (*e.g.* "Document 1"). This forced users to interact with the open documents to some degree (in order to determine

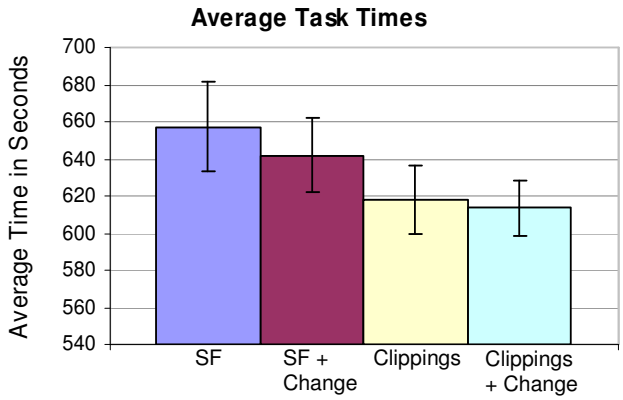

Figure 5: Average task time (*i.e.,* time to complete all tasks) per interface. Clipping List interfaces were significantly faster. Error bars represent standard errors.

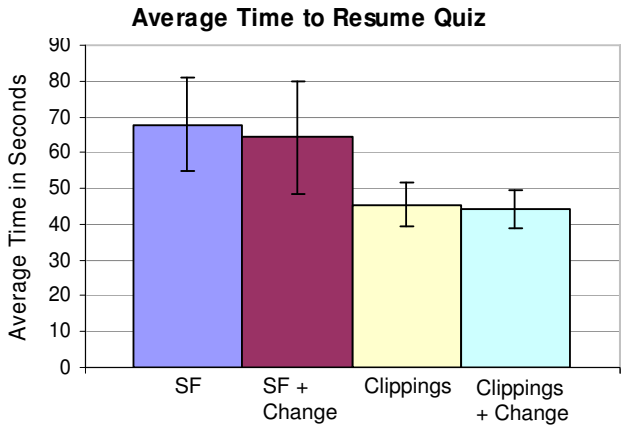

Figure 4: Average time to resume the Quiz task after updates per interface. Users reacted to updates quicker using Clipping List interfaces (without significance).

what document to upload next), which more closely modeled real work. The 2 emailed documents were named with their order number (*e.g.* "2.doc"). We programmatically timed emailed documents to arrive 30 - 120 seconds after their preceding document had finished uploading.

While waiting for documents to upload or for email to arrive, users were instructed to work on other tasks, but to return as soon as possible. After all 5 documents had been uploaded, a message to stop the task timer appeared and users would stop the Upload task timer.

Puzzle

Users were instructed to only work on the Puzzle task when they were waiting for both Quiz and Upload. Though we recorded the Puzzle task time, we stopped 8 users before they completed the puzzles in order to conclude the session in a reasonable amount of time and always made it clear to users that the Quiz and Upload tasks were higher priority.

This task included 4 windows: 2 with images and 2 with square pieces from the 2 images mixed together. Users first dragged pieces for each picture into one document. Then they rearranged the pieces to resemble the 2 model images.

The Puzzle task turned out to be extremely engaging, (as witnessed by users' focus on this task and by their comments after finishing it) probably because the pictures were designed to be challenging to reassemble. This was important since an engaging task would be more realistic and enable us to see bigger differences between users' ability to monitor the periphery with different peripheral interfaces.

Email

In addition to receiving task-relevant email (quiz answers and documents to upload, both from Jane Morgan), users received distracter email from different people on a variety of topics (again, in an effort to mimic the real world). Distracters were sent programmatically, at random rates between 20 and 90 seconds apart. Users received an average of 12 distracter emails per interface. They differentiated task-relevant email by the sender: Jane Morgan was the only person who sent email of importance. The presence of distracters enabled us to see if any interfaces performed poorly with the presence of uninteresting updates.

Interface Setup

The user interfaces being studied were presented on both the left side of the left monitor and the right side of the right monitor. Tasks were preset to include the correct windows and arranged in the periphery to allow maximal visibility of all windows. Window height and width were scaled to 25% of their original size (the default for Scalable Fabric). To maintain consistency across users, we disabled several Scalable Fabric functions during the study: clicking on a task name to restore or minimize all task windows, dragging windows into the periphery, moving windows in the periphery, and moving tasks in the periphery.

For Clipping Lists interfaces, we pre-selected Clippings. Since we designed the tasks and knew exactly how the tasks would be optimally completed, we were able to select the most useful Clippings. We believe that context awareness and document processing techniques could be combined to aid the user in at least semi-automating this task.

Measures

Dependent variables collected during the course of the study included task time, task resumption time, the number of task switches, the number of window switches within each task, user satisfaction ratings, and overall interface preference. Other than satisfaction ratings and preferences, all other measures were automatically collected via logging tools installed on the user's machine.

Design

The study design was a 2 (Semantic Content Extraction: Scalable Fabric v. Clipping Lists) x 2 (Change Detection: No Change Borders v. Change Borders) x 2 tasks, within subjects design. We counterbalanced the presentation order of all conditions and task sets across users (4 isomorphic

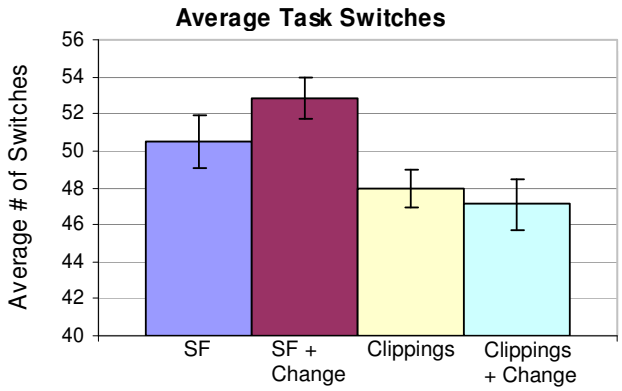

Figure 6: Average number of task switches per interface. Clipping List interfaces significantly reduced task switches; Change Borders increased them for Scalable Fabric.

task sets were rotated through the conditions so that users were not performing the same task in each user interface).

Procedure

Users were run in pairs with an experimenter present in the room. After greeting the users, the experimenter presented the overall study procedure and then walked the users through a set of practice tasks using their first interface (counter-balanced across users). Once users completed the practice tasks (approximately 30 minutes), they began the study proper. Users completed the Quiz, Upload, and Puzzle tasks using a single interface and then completed a satisfaction questionnaire about it before moving on to the next interface. In between each interface, we explained how to use the next interface.

At the end of the session, we debriefed the users, provided the software gratuities, and escorted them out of the building. Total session time was approximately 2 hours.

Results

We used a 2 (Semantic Content Extraction) x 2 (Change Detection) RM-ANOVA to analyze the data presented throughout this section, unless otherwise stated.

Task Times

As is standard practice for time data, all task times were transformed into log times in order to render the distributions normal to deal with skew and outliers in the original data. We analyzed the data for each of the priority tasks (Quiz and Upload) together. A data file for one user was missing due to logging errors.

We found significant main effects for the influence of Semantic Content Extraction, $F(1,25)=9.0$, $p=.006$. We did not observe any other significant main effects or interactions in the task time data.

These results show that having Clipping Lists, a form of semantic content extraction, in the periphery allowed users to perform their tasks significantly more efficiently (average time for Scalable Fabric = 649.7 seconds v. Clipping Lists = 615.6 seconds). Also, it shows that our implementation of change detection did not significantly improve users' efficiency. These results are shown in Figure 5.

Time to Resume

Again, we transformed the time to resume data to log times for analysis. We collected this data by measuring the time from which the user completed an update on the quiz or upload tools and when the user responded to the update by clicking on the tool.

We observed no significant main effects or interactions at the $p=.05$ level in this data. However, Figure 4 shows a trend toward a significant effect for the Quiz task only, $F(1,24)=2.7$, $p=.115$: interfaces with Clipping Lists improved resumption times, on average (Scalable Fabric = 66.0 seconds v. Clipping Lists = 45.0 seconds).

Task Switches

We counted a task switch as a user starting on a window in one task (*e.g.,* Upload) and then clicking on a window in a second task (*e.g.,* Quiz). Data from three subjects were missing due to logging errors.

We observed a significant main effect of Semantic Content Extraction, $F(1,23)=13.0$, $p=.002$, indicating that Clipping Lists reduced the number of task switches, enabling users to better maintain their task flow. These results are shown in Figure 6. We did not observe any other significant main effects or interactions in the task switch data.

We also examined task switches caused by distracter email, which indicate inopportune switches to the email task. To do this, we analyzed the proportion of switches to email that occurred after distracters arrived, over the total number of distracters received. Logging of switches due to distracters was affected by an error, so we lost small amounts of data from 11 users. Therefore, we chose not to include these users in this analysis. We report these findings anyway because the results are intriguing, but caution should be taken until they are replicated.

We found a significant main effect of Semantic Content Extraction, $F(1,15)=42.9$, $p<.001$, as well as Change Detection, $F(1,15)=6.3$, $p=.024$. This means that when Clipping Lists were present, users were significantly less likely to switch away from their current task due to spam email. When Change Borders were present, users were significantly more likely to switch away from their current task due to spam email. This negative effect of Change Borders was particularly strong with Scalable Fabric.

Window Switches

We analyzed the number of window switches for each priority task (Quiz and Upload) separately. Window switches were counted as the number of switches to a different window within a single task (Quiz or Upload). Data from three subjects were missing due to logging errors.

We found significant main effects for the influence of Semantic Content Extraction for both tasks, Quiz $F(1,23)=4.6$, $p=.042$, and Upload $F(1,23)=51.9$, $p<.001$. These results show that Clipping Lists significantly reduced the number of window switches within both priority tasks, which may improve task flow. These results are shown in Figure 7.

Satisfaction

We ran a 2 (Semantic Content Extraction) x 2 (Change Detection) x 13 (Questionnaire Item) RM-ANOVA on the satisfaction questionnaire ratings. We chose to not include surveys from two users who did not complete all questions.

We found significant main effects for Semantic Content Extraction, $F(1,24)=11.3$, $p=.003$, Change Detection, $F(1,24)=9.5$, $p=.005$, and Questionnaire Item, $F(14,336)=34.6$, $p<.001$. User interfaces with Clipping Lists were rated significantly better than interfaces without. In addition, interfaces with Change Borders were rated significantly better that interfaces without. All of the average satisfaction ratings for the user interfaces in the study are shown in Figure 8.

When asked to choose their preferred interface, 17 out of 25 users who responded chose Clipping Lists with Change Borders (significant by chi square test), 4 chose Scalable Fabric with Change Borders, 2 chose Clipping Lists, and 2 chose the baseline Scalable Fabric.

User Comments

User comments supported and explained our quantitative results. They explained why the scaling used in Scalable Fabric needed improvement. In particular, they needed more obvious visual cues that tasks had updated (such as Change Borders) and clearer identifying information visible for windows in order to recognize them. One user said, "It was impossible to know when something was ready to use and I had to open all the windows for a particular project, to tell which window contained the item I needed. It just didn't feel efficient." Users also needed more context about a task in order to switch to the correct task: "[For] text based documents… it is better to be able to just read a little bit of the text, because that gives enough context information to be able to switch to the proper task."

Users explained why Change Borders were useful, "…the border made monitoring the two main tasks pretty easy and didn't require a lot of mental action." They also explained why Change Borders caused problems for email, "…the majority of emails are irrelevant to the tasks being performed here, and therefore will have a negative effect if [I] check email every time the color changes." Users thought frequent emails were realistic: "In my real-life job… [I] can expect to receive an e-mail every few minutes."

User feedback favored the Clipping List interfaces. They thought Clippings made it easier to distinguish documents, determine to which task or window to switch, and improved monitoring ability by exposing updating content (which they especially liked for email). One user said, "This display allowed me to see all of the data that I needed at a glance."

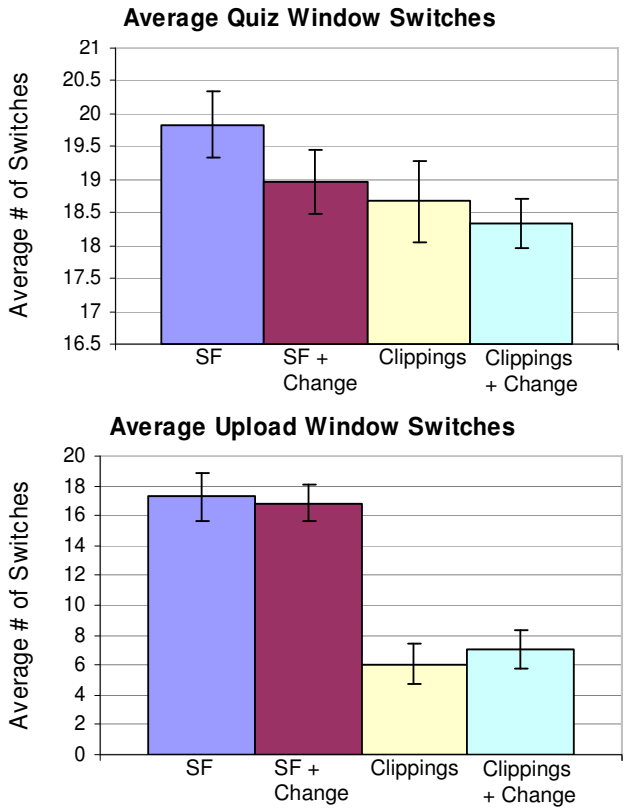

Figure 7: Average number of window switches per interface for the (top) Quiz and (bottom) Upload tasks. Clipping List interfaces significantly reduced window switches.

Additionally, users pointed out a few improvements that could be made. One suggested that the area under the mouse on shrunken windows enlarge as if the mouse were "a magnifying glass." One user wanted "something more obvious than the green border, to show a task complete." Another user wanted the Clippings to be interactive: "the e-mail [Clipping] is… difficult to follow. I would rather be able to scroll the peripheral display to a particular message and click on that to open the desired e-mail." A few users thought the aesthetics of the Clipping Lists could be improved with "more uniformity in the... text type," and by making all the Clippings the same width.

DISCUSSION

Our results enable us to examine the effects of our implementations of scaling, change detection, and semantic content extraction on task flow, task resumption timing, and reacquisition.

Maintaining Task Flow

Users switched tasks significantly less often with Clipping Lists interfaces, which used semantic content extraction, indicating that it improved task flow. We informally ob-

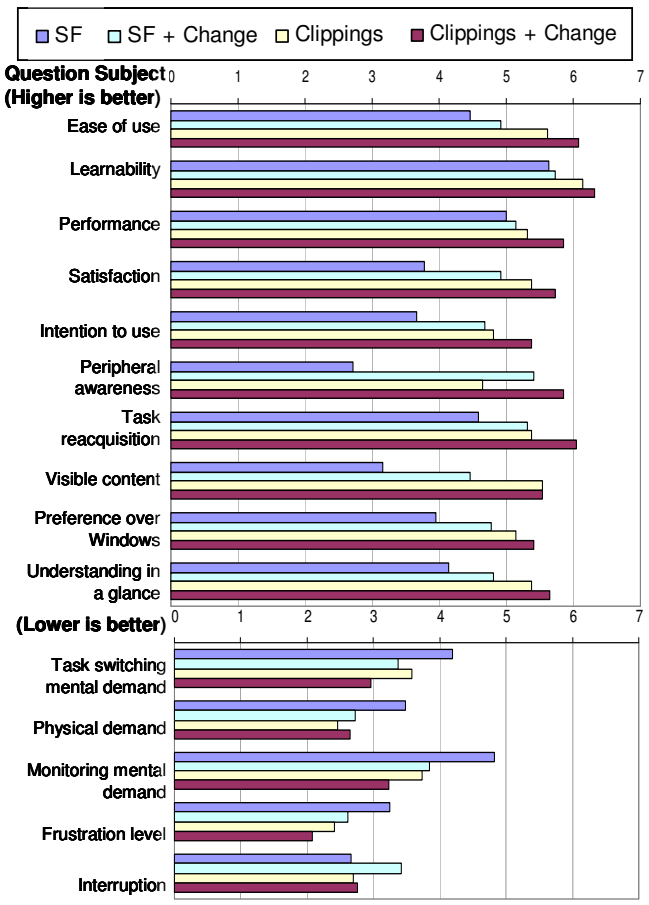

Figure 8: Survey result averages (Likert scale 1-7). UIs with Clipping Lists performed significantly better than those without, as did interfaces with Change Borders.

served that users glanced more quickly at Clipping Lists and returned to their focal task, implying less detriment to task flow. We feel adding Clipping List-style interfaces to peripheral displays could significantly enhance information workers' ability to remain in task flow, while efficiently responding to important task updates

Knowing When to Resume a Task

Our results showed that the Clipping Lists interfaces, using semantic content extraction, were most effective at making users aware of when to resume a paused task. First, task times were significantly faster for Clipping Lists, indicating more efficient multitasking. Though not significant, resumption times for Clipping Lists were faster, suggesting that users might have known when to return to tasks—a trend that deserves further attention. Finally, task switches were significantly fewer, implying that users did not have to manually check if peripheral tasks were ready.

Change Borders (a version of change detection), when used without Clipping Lists, had a negative effect on users' ability to resume email at appropriate times. From user feedback, it is clear that the one bit of information was not enough; users needed more information to determine if an email was important.

Having relevant task information drawn out in Clipping Lists enabled these effects. This suggests that semantic content extraction is important for giving users enough information to know when to resume a task.

Task Reacquisition and Recognition

Clipping Lists (semantic content extraction) were also most effective at enabling users to easily get back into tasks and recognize task-relevant windows. Qualitatively, users told us that Clippings made it easier to distinguish between windows. Quantitatively, window switching results showed that Clipping List interfaces significantly reduced the number of window switches within both priority tasks. This indicates that Clipping Lists better enabled users to recognize windows and get back to resumed tasks. We informally observed that users seemed to spend less time staring at peripheral tasks when switching.

Design Implications and Future Work

Our results show that Clipping Lists were effective and users told us this was because they provided the most relevant task information (*i.e.,* they used semantic content extraction). Clipping Lists improved task times by 29 seconds on average; Clipping Lists with Change Borders improved task times by 44 seconds on average. These task switching improvements are cumulative, adding up to a sizeable impact on daily multitasking productivity.

Presumably, the higher detail shown in Clippings required more attention to process, but Clipping Lists were still more efficient than scaling and Change Borders. Peripheral display designers have focused much effort on highly abstracted information and non-text based UIs. The fact that a detailed portion of a window, as long as it shows relevant information, offers improved task performance is quite interesting. Though liked by users, Change Borders did not provide as much of a performance benefit. We assert this was because change detection did not provide enough information to help users recognize windows, reacquire tasks, or to determine if an update was worthy of interruption. Specifically, more relevant task information is needed.

Unfortunately, at this time, semantic content extraction requires extra effort from users (*e.g.,* to select relevant task information) or from designers (*e.g.,* to design peripheral elements with pre-selected data). Given that semantic content extraction is more effort for either users or designers, is it worth it? We argue that the results of this study suggest it is. Results showed significant performance benefits as well as user preference for interfaces using semantic content extraction. Neither scaled windows nor change detection seemed to provide the right information to improve multitasking performance. Extracting relevant task information seems a crucial part of designing peripheral displays for multitasking, and clipping small pieces of window content proved an effective way to semantically extract content.

These results help us direct future design and research efforts. Because semantic content extraction is so important, we will direct more effort toward developing interfaces and algorithms for selecting relevant task information from windows. To automate some of this burden, more research is needed to correctly extract relevant content. In addition, since we knew how to best perform the study tasks and extracted optimal content in our study, we cannot comment on how this would generalize to less well-constructed or less predictable tasks. More research is needed to explore and identify relevant content in a diverse set of real-world tasks.

CONCLUSION

We set out to improve multitasking efficiency, focusing on helping users maintain task flow, know when to resume tasks, and more easily reacquire tasks. In an empirical study, we compared four peripheral interfaces using different types of abstraction that provided varying types of task information: scaling (showing a window's layout overview), change detection (whether or not a change had occurred), and semantic content extraction (displaying a small piece of the most relevant window content).

The main contribution of this paper is a set of results showing that semantic content extraction is more effective than both change detection and scaling in improving multitasking efficiency. We also show that semantic content extraction significantly benefits task flow, resumption timing, and reacquisition. These findings provide a better understanding of how to design peripheral displays that aid people who multitask, so that they focus their cognitive resources on the task at hand, instead of on task and windows management.

ACKNOWLEDGMENTS

We are grateful to Greg Smith and Brian Meyers for their technical expertise. We also thank Krzysztof Gajos, Gonzo Ramos, Amy Karlson, and Johnny Lee for study assistance.

REFERENCES

1. Czerwinski, M. and Horvitz, E. "Memory for daily computing events." *People and Computers XVI, Proc. of HCI*, 230-245, 2002.
2. Czerwinski, M., Horvitz, E., Wilhite, S. "A diary study of task switching and interruptions." *Proc. of CHI*, 175-182, 2004.
3. Ellis, J. & Kvavilashvili, L. Prospective memory in 2000: Past, present and future directions. *Applied Cognitive Psychology, 14*, 1-9, 2000.
4. Gonzalez, V.M., Mark G. "'Constant, constant, multitasking craziness': managing multiple working spheres." *Proc. of CHI*, 113-120, 2004.
5. Grudin, J. "Partitioning digital worlds: focal and peripheral awareness in multiple monitor use." *Proc. of CHI*, 485-465, 2001.
6. Gwizdka, J. "Timely reminders: a case study of temporal guidance in PIM and email tools usage." *Extended abstract of CHI*, 163-164, 2000.
7. Henderson, D.A. Jr. and Card, S. "Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface." *ACM Transactions on Graphics, 5, 3*, 211-243, 1986.
8. Hutchings, D.R., Stasko, J. "Shrinking window operations for expanding display space." *Proc. of AVI*, 350-353, 2004.
9. Jones, W.P., Bruce, H. and Dumais, S.T. "Keeping found things found on the web." *Proc. of CIKM*, 119-126, 2001.
10. Lamming, M.G. and Newman, W.M. "Activity-based information retrieval: technology in support of personal memory." *Information Processing, 92, 3*, 68-81, 1992.
11. MacIntyre, B., Mynatt. E.D., Voida, S., Hansen, K.M., Tullio, J., Corso, G.M. "Support for multitasking and background awareness using interactive peripheral displays." *Proc. of UIST*, 41-50. 2001.
12. Matthews, T., Rattenbury, T., Carter, S., Mankoff, J., Dey, A. "A peripheral display toolkit." *U.C. Berkeley Tech Report, CSD-03-1258*, 2003.
13. Mark, G., Gonzalez, V.M., Harris, J. "No task left behind?: examining the nature of fragmented work." *Proc. of CHI*, 321-330, 2005.
14. O'Connail, B. and Frohlich, D. "Timespace in the workplace: dealing with interruptions." *Extended Abstracts of CHI*, 262-263, 1995.
15. Pedersen, E.R., Sokoler, T. "AROMA: abstract representation of presence supporting mutual awareness." *Proc. of CHI*, 51-58, 1997.
16. Plaue, C., Miller, T., Stasko, J. "Is a picture worth a thousand words?: an evaluation of information awareness displays." *Proc. of GI*, 117-126, 2004.
17. Robertson, G. Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D.R., Meyers, B., Robbins, D., Smith, G. "Scalable Fabric: flexible task management." *Proc. of AVI*, 85-89, 2004.
18. Schneiderman, B., Bederson, B.B., "Maintaining concentration to achieve task completion." *Proc. of DUX*, To Appear, 2005.
19. Sellen, A.J., Louie, G, Harris, J.E. and Wilkins, A.J. "What brings intentions to mind? An in situ study of prospective memory." *Memory, 5(4),* 483-507, 1996.
20. Tan, D.S., Meyers, B., Czerwinski, M. "WinCuts: manipulating arbitrary window regions for more effective use of screen space." *Proc. of CHI*, 1525-1528, 2004.
21. Terry, W.S. "Everyday forgetting: data from a diary study." *Psychological Reports, 62*, 299-303, 1988.