SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 28(3), 225–248 (MARCH 1998)

Elastic Windows: Design, Implementation, and Evaluation of Multi-Window Operations

eser kandogan[1] and ben shneiderman[2]
[1]*Department of Computer Science, Human-Computer Interaction Laboratory,* [2]*Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, U.S.A.*
(email: {kandogan,ben}@cs.umd.edu)

SUMMARY

Most windowing systems follow the independent overlapping windows approach, which emerged as an answer to the needs of early computer users. Due to advances in computers, display technology, and increased information needs, modern users demand more functionality from window management systems. We propose Elastic Windows with improved spatial layout and rapid multi-window operations as an alternative to current window management strategies. In this approach, multi-window operations are achieved by issuing operations on window groups hierarchically organized in a space-filling tiled layout similar to TreeMaps.[1] Sophisticated multi-window operations have been developed to handle fast task-switching and to structure the work environment of users to their rapidly changing needs. We claim that these multi-window operations and the tiled spatial layout dynamics decrease the cognitive load on users by decreasing the number of window operations. This paper describes the Elastic Windows interface in detail and then presents a user study conducted to compare the performance of 12 users with Elastic Windows and traditional Independent Overlapping Windows. User performance was measured in terms of task environment setup, switching, and task execution for 2, 6, and 12 window situations. Elastic Windows users had statistically significantly faster performance for all tasks in 6 and 12 window situations. These results suggest promising possibilities for multiple window operations and hierarchical nesting, which can be applied to the next generation of tiled as well as overlapped window managers. © **1998 John Wiley & Sons, Ltd.**

key words: window management; multi-window operations; hierarchical windows; personal role management; World Wide Web; windowing system evaluation

INTRODUCTION

Window management strategies can only be analyzed by a careful consideration of the tasks for which windows are used. Card *et al*. attempted to categorize tasks by the functions provided by windows, which they listed as:[2]

(a) More information.
(b) Access to multiple sources of information.
(c) Combining multiple sources of information.
(d) Independent control of multiple programs.
(e) Reminding.

CCC 0038–0644/98/030225–24$17.50

Received 4 November 1996
Revised 14 May 1997
Accepted 8 September 1997

(f) Command context/active forms.
(g) Multiple representations.

Most current windowing systems follow the independent overlapping windows approach, which emerged as an answer to the needs of early computer users. These windowing systems no longer provide efficient means to serve the functions in this list. With advances in computer networks, the Internet, and Web browsers, users are collecting not only more information but also different formats such as image, video, sound, structured text, etc. This increase in the amount and variety of information creates problems in accessing and using stored information when current strategies are employed.

In early computer systems, in order to access information users had to recall exact file names, directory structures, etc. These attributes are called nominal attributes of information. Recalling nominal attributes becomes more difficult as the amount of information increases, because nominal attributes refer to computer-based objects rather than task-based objects.

With the introduction of windows, users can use spatial attributes like location of icons or windows to access information. However, current systems provide limited capabilities in terms of icon and window placement, generally a single screen where icons and windows can be placed independently anywhere on the screen. As a result, computer screens become cluttered and windows are hidden making it harder to access information using spatial attributes, a situation called Windowitis by Kahn *et al*.[3] They observed that in such situations users become disoriented, lose the relationships that exist between windows due to loss of spatial cues, and become unproductive in completing their tasks. Increased clutter makes it hard for users to recognize visual cues on the screen as reminders of their unfinished tasks.

In current systems, visual attributes for information access are underutilized. As a result, the computer screen fills up with many similar icons with only the label under the icon being different. Therefore, users have to scan all the icons on the screen in two dimensions rather than recalling spatial or visual attributes of information. Although most of the applications set their icon image, it is usually difficult for users to set different visual primitives for pieces of information they collect. Automatically generated icons, dynamically reflecting attributes of information, might be helpful. We believe that users performance for information access can be improved by incorporating visual attributes of information into windowing systems.

In current systems tasks that require interaction with information contained in multiple windows require the user to operate one window at a time. Providing multiple window operations with simple actions is likely to help users.

Traditionally, windows have been used mostly for a single application. In that sense, multiple windows functioned as independent control of multiple programs, by distributing applications into multiple windows. There is now a growing tendency for applications to have multiple windows, where functionality is distributed into multiple windows as opposed to applications. The danger is that windows belonging to different applications can be mixed, when users are working on multiple tasks.

Current user interfaces exploit visual metaphors of physical objects, such as documents and folders on a desktop. Applications are confined to overlapping rectangular windows on the desktop, similar to their physical counterparts (papers, calendar, calculator, etc.).

Novel approaches emphasize a docu-centric approach (Microsoft OLE and Apple's Open-Doc) in which documents become more important and applications fade into the background. Information objects of various types such as text, image, video, sound, spreadsheet cells are organized in documents independent of applications. However, these approaches lack an effective scalable organization method.

Although these innovations are one step to achieve a visual environment in harmony with users' perceptions of their work, an effective organization of information according to users' roles that reflects this perception is needed.[4,5] Organization criteria should be the users' roles, rather than documents. Users can have a number of different roles possibly multi-level, and overlapping.

The key to personal role management is organizing information according to the roles that an individual has in an organization. It complements the previous approach, where information is distributed into multiple windows, by providing an effective organization based on users' roles. When users are working in a role, they have the most relevant objects regarding that role like schedules, documents, correspondence with people, etc. all visually available. These visual cues remind them of their goals, related individuals, required tasks, and scheduled events all within the context of the current role. Users should be able to create/abandon roles, extend/modify the current role hierarchy by inserting/deleting sub-roles, and combine roles and/or objects within different roles.

Our earlier work[6] stated the requirements for future windowing systems. A more complete list is as follows:

(a) Support a unified framework for information organization and coordination according to users' *roles*.
(b) Provide a visual, spatial layout that matches *semantics*.
(c) Support *multi-window* operations for fast arrangement of information.
(d) Support information access with *partial knowledge* of its *nominal*, *spatial*, *temporal*, and *visual* attributes and relationships to other pieces of information.
(e) Allow fast *switching* and resumption among roles.
(f) Free users' cognitive resources to work on *task domain operations* rather than computer domain operations.
(g) Use *screen space efficiently* and productively for tasks.
(h) *Increase collaboration within the window manager* to support more powerful information transfer schemes among multiple windows.

In the next section, we give a detailed description of the Elastic Windows approach followed by related work. Next, the user study comparing performance of Elastic Windows to traditional Independent Overlapping Windows is described in detail along with the results and observations made.

ELASTIC WINDOWS

Principles

Elastic Windows design is based on three principles: *hierarchical window organization*, *multi-window operations*, and *space-filling tiled layout*.

Hierarchical window organization

Hierarchical window organization supports users structuring their work environment according to their roles. It allows users to map their role hierarchy onto the nested rectangle tree structure. Figure 1 displays the hierarchical organization of different roles of a university professor. This professor is advisor to a number of students in a number of research projects, teaches two courses this semester, is liaison to three companies, and has personal duties.

The hierarchical layout clearly indicates the hierarchic relationship between the contents of the windows by the spatial cues in the organization of windows. It provides the users with an overview of all their roles, where they can pick any role or parts of it and start working on it.

Problems in accessing and managing large collections of information are reduced by establishing hierarchical nested windows for the organization of information objects (Figure 2). Grouping of windows provides role-based context by bringing the set of windows associated with a role into one overall context. In Elastic Windows, window hierarchies can be collapsed into a single icon (or other visual primitives) which allows manipulation (e.g. move, copy, open, close, pack, etc.) of a collection of objects as a single entity. This form of visual aggregation also reduces access

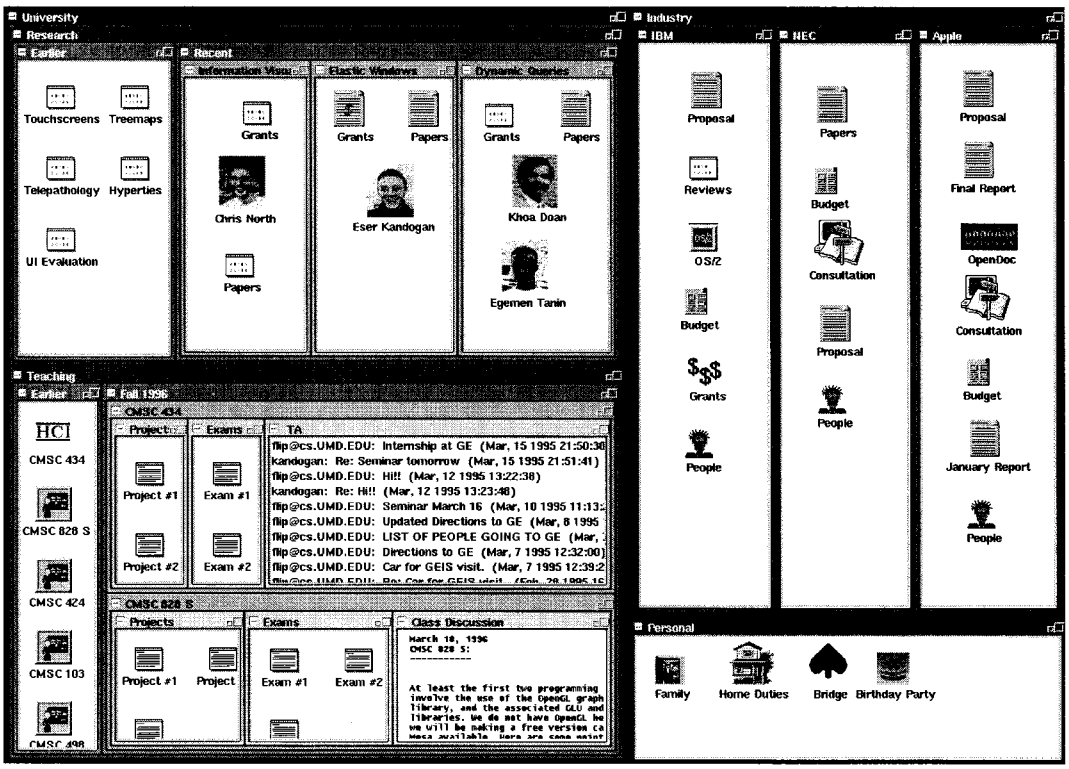

Figure 1. Personal role manager: hierarchical organization of a university professor's roles

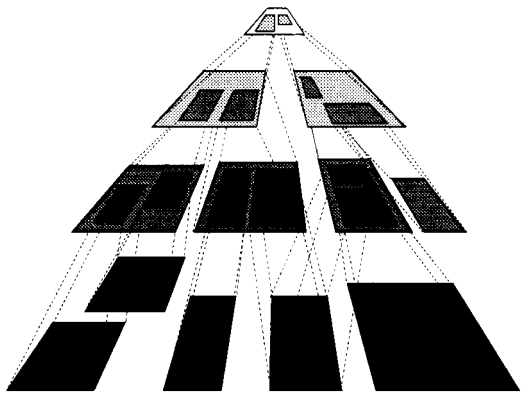

Figure 2. Hierarchical nested windows

and management problems of large collections. It also saves screen space allowing users to concentrate at any level of detail.

Current window management strategies have a limited notion of workspace. Most systems provide only one screen, whereas more novel systems such as Rooms[7] and UNIX vtwm-clone window managers provide multiple virtual screen spaces where windows can be placed in any of these spaces. Some systems also provide an overview where users can look at thumbnail images of the screen layouts and use the overview to switch to these spaces and to move windows among spaces. Basically, these two approaches allow only two-levels of workspace.

In Elastic Windows, users can work at any level of detail regarding their roles. Multi-level task focus is provided by allowing users to make any window full screen at any point in the hierarchy. This is a multi-level approach as opposed to the two-levels (overview and workspace) in most virtual screen window managers.

Multiple window operations

Typically, people organize papers on their desk as piles, and move all of them simultaneously. Malone[8] found that users like to group items spatially. We claim that providing multi-window operations on groups of windows can decrease the cognitive load on users by decreasing the number of window operations.

In Elastic Windows, multiple window operations are achieved by applying the operation to a group of windows at any level of the hierarchy. The results of the operation are propagated down to windows inside that group recursively. This way groups of windows can be packed, resized, or closed with a single operation.

Operations like multi-window open, close, resize, pack, and unpack enable users to change the window organization quickly to compare, filter, and apply the information. Operations like save, load, and change layout work on multiple windows allowing users to quickly restore previous work environments as well as try alternative layouts for different subtasks within the same context. Another way to achieve multi-window operations is to select an operation and apply it to a number of windows rapidly in a serial manner.

Space-filling tiled layout

We have taken a space-filling tiled approach similar to TreeMaps[1] at this stage of our research to explore its potential for productive use of screen space. Non-overlapping approaches may have an advantage in that they avoid wasted space and disturbing overlaps. On the other hand, in space-filling tiled approaches, window contents may not always conform to different window sizes and small window sizes may not provide sufficient visibility of window contents.

In Elastic Windows, groups of windows stretch like an elastic material as they are being resized, and other windows shrink to make space. The effect of changes in window size on the contents depends on the application. Contents can be scaled, clipped, reformatted, or aggregated (such as showing only section headings of a document). For example, upon shrinking a window used for editing a document, it might be preferable to keep the same magnification factor; but when viewing a web document, scaling of contents might be preferable.

We have chosen the tiled window layout in order to maximize the visibility of windows. People typically try to organize windows to be non-overlapping while working on a task, even when overlapping windows are allowed. As Cohen *et al.*[9] stated, overlapping window layouts are difficult to handle when large numbers of windows must all be visible at once, and they come and go rapidly.

In Elastic Windows, hierarchies of windows are represented by the borders surrounding the sub-windows. Users are quite flexible in the placement of sub-windows in a group window. There is no strict horizontal or vertical placement rule within window groups.

Initially, we developed a mail-tool application based on these three principles. It has been a test-bed to try hierarchical organization of windows to help users manage large collections of e-mail messages, multiple window operations for reducing the number of window operations, and space-filling tiled layout for efficient management of screen space. Figure 3 displays the mail-tool written using Elastic Windows principles. The new messages are shown iconized in the left window. Old messages from people at the University of Maryland, and at Bilkent University, from friends, and association members are grouped on the right in the OldMail window, each in separate windows with further classification. The layout provides the user with an overview of all correspondence, where users can pick any category and work on it.

The potential of hierarchical organization of windows as an overview led us to provide hierarchical contexts for interaction with information organized according to user's roles. We then developed Elastic Windows for personal role management to provide users a uniform framework for management of personal information by including information formats such as documents, images, lists, etc. besides e-mail messages. In Figure 1, the hierarchical organization of different roles of a university professor is shown. We also introduced aggregation of information by allowing users to collapse a hierarchy of information into a single visual primitive for efficient management of information.

Recently, WorldWide Web (WWW) browsing capability was added into Elastic Windows. Figure 4 shows a screen snapshot of Human-Computer Interaction Lab (HCIL) web pages. The Research Project Descriptions link is opened below the HCIL main page on the left. All six current project descriptions are opened with a single select and open action, grouped in a container window on the right. Elastic

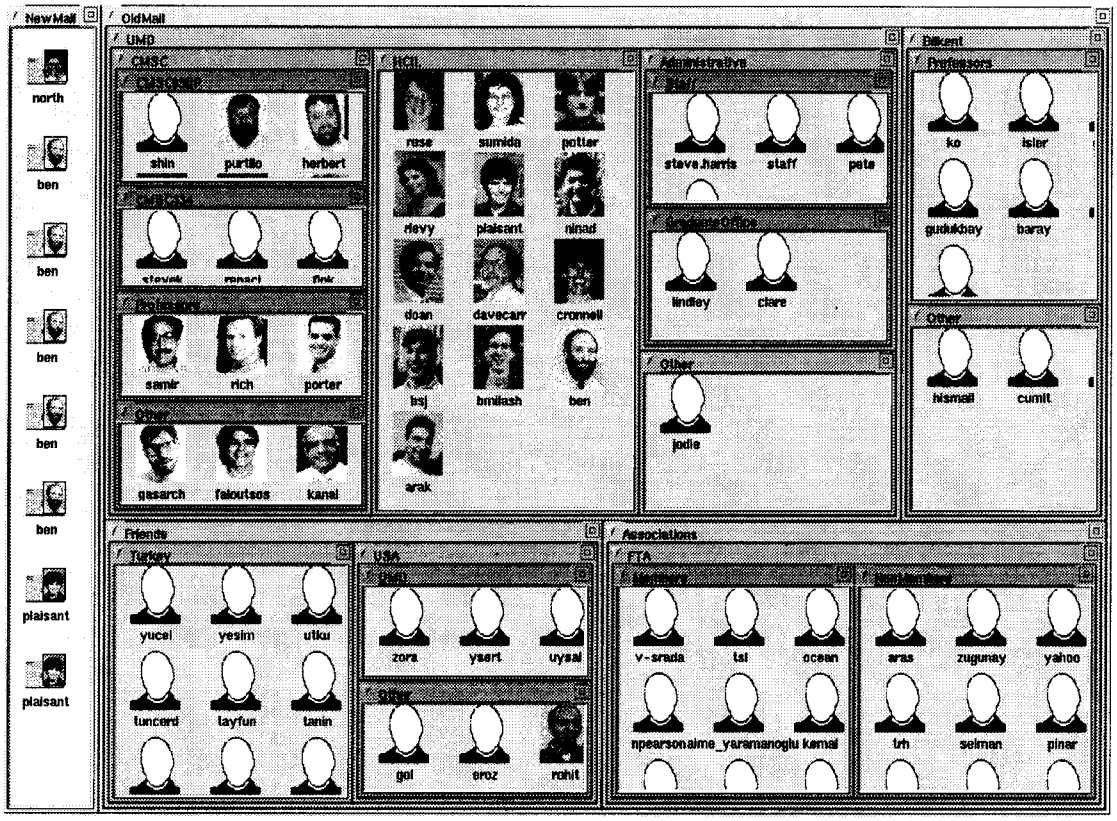

Figure 3. Mail-tool: organization of correspondence in a hierarchical layout gives an overview

Windows principles support a variety of WWW browsing strategies such as hierarchical browsing, multi-page open, comparison, and filtering. It allows users to keep a visual history of browsing, to explore multiple pages simultaneously by a multiple-open operation, and to organize visited pages hierarchically on a two-dimensional layout, facilitating spatial and visual access, as an alternative to a bookmark list.

Layout dynamics

Due to the space-filling tiled nature of the layout, window size changes affect size of other windows in the same group. In Elastic Windows the proximity of effect is limited only to windows under the same group and their sub-windows.

Effect of the changes in the window size under the same group is split proportionally according to the window sizes. Depending on the border dragged and the direction of drag, it results in either a push or pull as shown in Figures 5(a)–(c). In both of these cases, window sizes are updated proportionally to their previous sizes as calculated below, where *width* and *width′* stand for the width of the windows before and after the push of windows C and D by $\triangle$ respectively.

In Elastic Windows size changes in the upper levels are propagated down to their subwindows recursively. Figure 6 shows an example of resizing the Teaching window

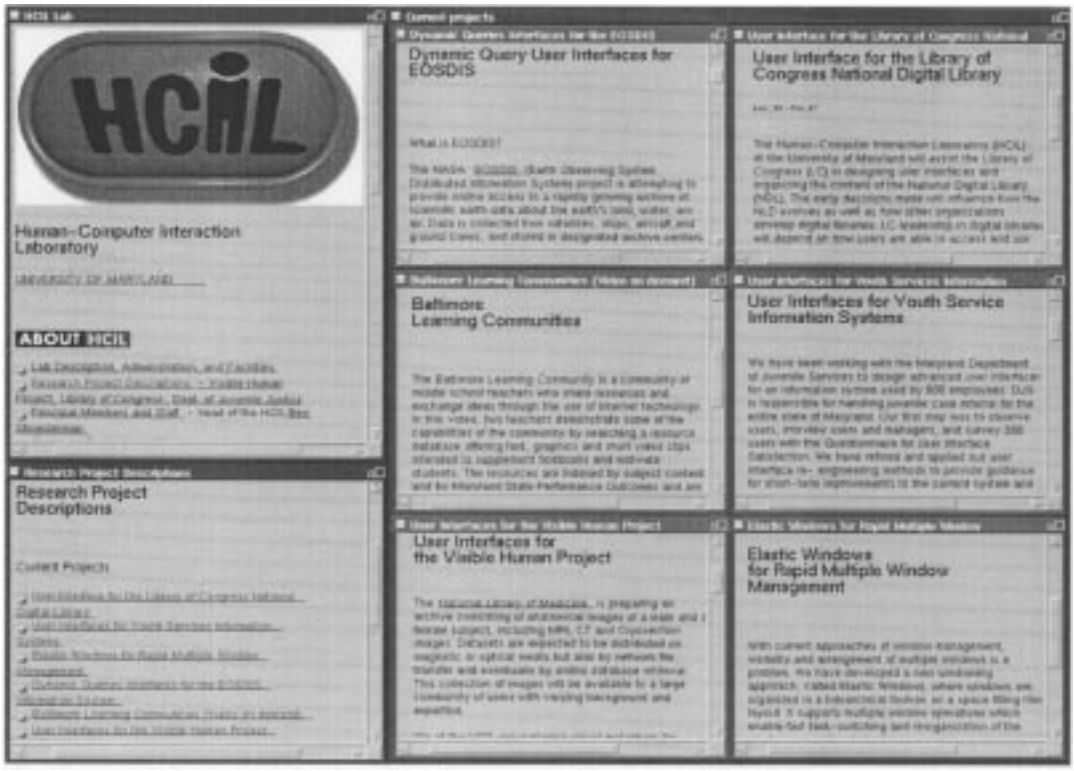

Figure 4. WWW browser: HCIL homepage with descriptions of six current projects

Figure 5. (a) Original layout, (b) window B pushes windows C and D, (c) window B pulls windows C and D

(original layout as shown in Figure 1), pushing the surrounding windows to the sides proportional to their sizes. Each Elastic Window has a default minimum window size, but users can also set a different value for each window.

Figure 6. Teaching window resized pushing surrounding windows to the sides (from Figure 1)

Visual access

In the Elastic Windows approach, windows or a hierarchy of windows can be iconified. When windows are iconified users can change the icon image by a simple image editor capable of loading standard image formats. For a hierarchy of windows, however, users can set their icons to be *hierarchicons* where the icon image reflects the window hierarchy. This miniature image of the hierarchy helps users in recognizing the icon among many. Moreover, users can click on different regions of the hierarchicon to select windows at different levels. An example series of actions for hierarchicons is given in Figure 7.

Window borders are used to indicate hierarchical groupings of windows. Border coloring gradually changes according to the level of the window in the hierarchy

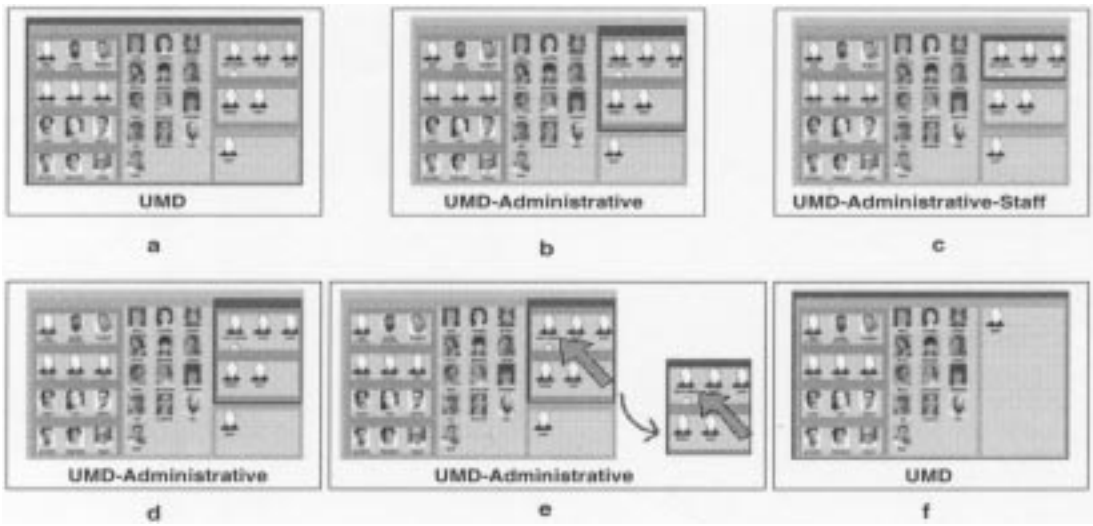

Figure 7. Hierarchicons. (a) Initial, top level hierarchy selected, (b–c) down in the hierarchy, (d) up in the hierarchy, (e) selected subhierarchy dragged, (f) changes are reflected in the hierarchicons image

thus making groups recognizable. Also, users can set the color of the window backgrounds, this makes windows easily recognizable. Background color of a higher-level window is propagated down the hierarchy (Figure 1).

Window operations

Multiple open/close operations

In Elastic Windows, selecting, dragging and dropping multiple objects (e.g. e-mail subject list, e-mail message, text icon, single window, hierarchy of windows, etc.) on the border of a window opens separate windows for each object in the selection. An empty window can also be opened as a 'container' window by double-clicking on a window border.

The position of the new window is determined by the border of the existing window. For example, clicking/dropping on the right border of a window will open a new window to the right of the current window pushing it to the left (Figure 8(a–b)).

A window is closed by selecting the Close operation from the menu. When a window is closed, the freed space is distributed to other windows at the same level

Figure 8. (a) Original layout, (b) five e-mail headers selected and empty container window is opened, (c) window group created with five e-mail messages

proportional to their previous sizes. The Close operation can also be applied to windows at any level of the hierarchy. Closing a higher level window will close all its sub-windows as well.

Creating window groups and hierarchies

Window groups can be created by opening a container window and dragging and dropping selected objects inside this window. A separate window is opened for each object in the selection as a member of the group surrounded by the container window borders. In Figure 8, five e-mail messages are selected from a list of message subjects, and opened in separate windows in the container window on the right. Multiple objects can be added or removed from the container window at any time. It is also possible to open a new empty container window within an existing container window to create hierarchical windows.

Multiple maximize operation

Users can focus on a set of windows and maximize them to cover the whole screen. This is particularly useful when users are expecting to work on a set of windows for a long time. With the maximize operation, users can use more of the screen real estate, avoiding the loss of screen space due to nesting.

In Elastic Windows users can work at any level of detail regarding their roles (Figure 9). Multi-level task focus is provided by allowing users to make any window full screen at any point in the hierarchy, giving users more space to focus on the task in that context.

Layout save/load/change operations

In Elastic Windows multiple layouts can be saved for each object, and loaded at a later time. This operation helps users to reconstruct their previous working environments easily. Users are also provided with three standard layouts: horizontal, vertical, and tiled.

Users can also save custom-made layouts for each object under different names and load them. This gives them flexibility in using alternative layouts for different subtasks within the same context without the burden of recreating the layouts used earlier.

Multiple resize operations

Windows at any level of the hierarchy can be resized by dragging on the border. All of the sides and corners of the window borders can be used in resizing. The drag direction and the border being dragged determines the effect as explained in the layout dynamics section. The corners of a window are used for diagonal resizing, whereas the sides are used for either horizontal or vertical resizing depending on the border.

Inside a group window with many windows open, typically users need to focus on one of these windows for a certain time. Bidirectional resizing allows resizing of a window in both directions by pushing/pulling the opposing borders by the same amount.

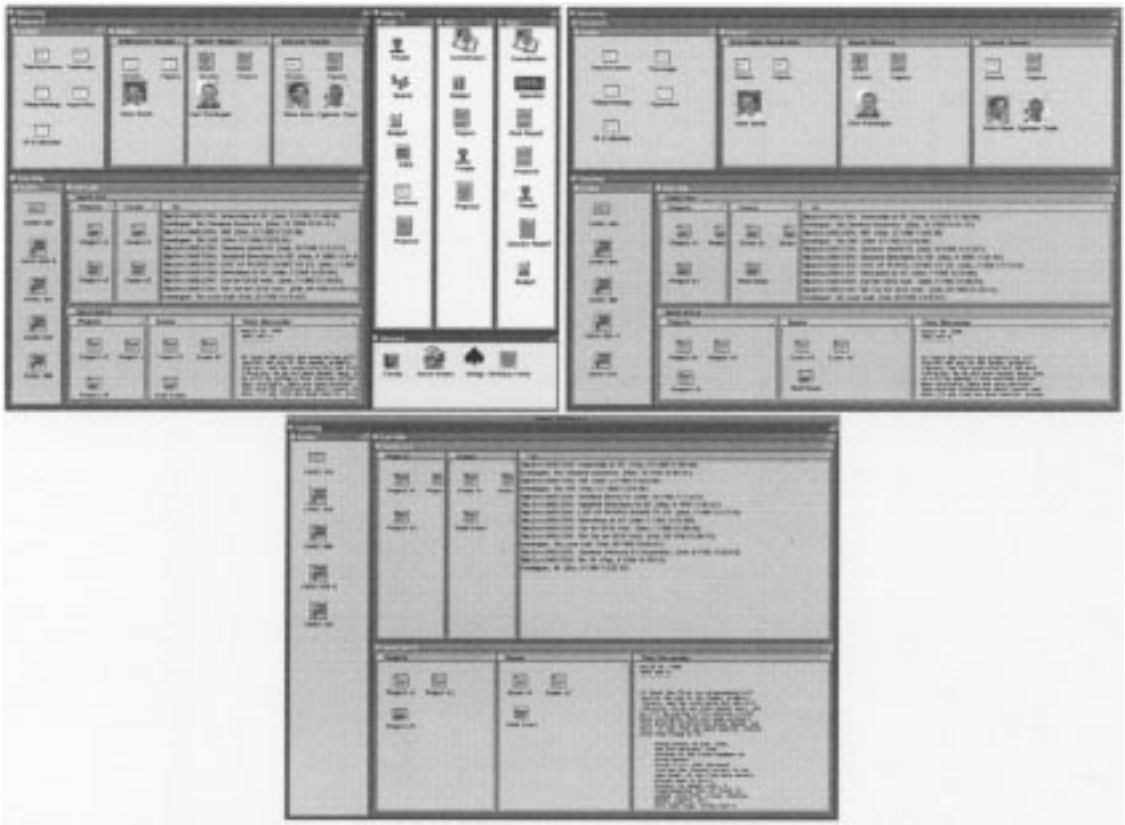

Figure 9. (a) Original layout as in Figure 1, (b) university window maximized, (c) teaching window maximized

Multiple copy/move operations

The copy operation creates a duplicate of a window or hierarchy of windows, with their contents, at a different position in the hierarchy. The proportions of the sub-windows are kept the same at the new position, but their sizes might change. Both the copy and the original access the same source, thus changes to the contents in one of them affect the other one as well. The move operation, however, relocates a window or hierarchy of windows to a new position in the hierarchy, while removing the windows at the old position.

Multiple pack/unpack and iconify operations

Windows at any level of the hierarchy can be packed by selecting from the menu. Packed windows appear in the same location, preserving the spatial cues with reduced size, but only their title is shown as a thin rectangular bar appropriately placed in the layout. Keeping the packed windows in the same position avoids the spatial disorientation and helps users later to locate them easily as well as to remind them of unfinished tasks.

Pack and unpack operations on groups of windows help users to filter-out unnecess-

ary information. When a packed window is unpacked all the windows in the group are restored to their previous sizes with a single action, so that users can reconstruct their previous working environments easily. The pack/unpack operations are primarily used to abandon a task for a while and open up space for other tasks. The purpose of an iconify operation, however, is to save the window layout and their contents for later uses.

Other operations

Users can set the minimum width or height of a window in order to protect the window from an unintentional resizing. This constraint on the window width or height can be removed by the corresponding release operation.

Multiple window operations can also be achieved by serial application of a window operation to a number of windows. Once the users selects an operation, it can be applied to a number of windows by clicking on the window.

The Elastic windows approach satisfies most of the requirements of future windowing systems as discussed in detail in the previous sections. In summary, we believe that hierarchical window organization provides a framework for information organization and coordination according to user's roles by mapping their role hierarchy onto the nested rectangle structure. While the hierarchical structure depicts a possible relationship among information contained in multiple windows, it also allows multi-window operations that permit fast task initiation, switching, and screen layout arrangement. Although information is hierarchically organized, information can also be accessed using incomplete knowledge of users on its nominal, spatial, and visual attributes. The space-filling layout of Elastic Windows might increase screen space utilization by avoiding wasted space among windows.

RELATED WORK

The Rooms system[7] uses multiple virtual workspaces, where the overlapping window strategy is used in each of these single-screen workspaces. Each task is devoted to a workspace, where users can switch to other tasks using either the overview or the doors between workspaces for rapid transitions. There is no mechanism which allows multi-window operations.

The Cedar[10] system also uses tiling, where windows are organized in two columns with an arbitrary number of windows in each column. It also uses a space-filling tiled layout, but proportional resizing is not provided. Windows can not be grouped hierarchically and multiple window operations are not provided. A number of other early windowing systems also use tiling. Myers has an excellent taxonomy of these early windowing systems.[11]

A more recent system, the Dylan programming environment, uses a pane-based window system,[12] which allows both horizontal and vertical panes, with a mechanism to create links between panes. However, it does not support multiple window operations and hierarchical organization of windows.

Recent research in more advanced information management user interfaces has generated a handful of interesting innovations. The Web-Book work at Xerox extends the 2D desktop metaphor to a 3D office metaphor.[13] Web documents can be organized hierarchically in metaphors like book, shelf, table, etc. While the approach facilitates

hierarchical organization of documents, multiple window operations are not supported. Although 3D metaphors may be appealing, we wonder whether the screen space is underutilized.

Pad++ introduced a novel technique for spatially organizing information on an infinitely zoom-able surface.[14] Web documents are presented in a two-dimensional tree structure at varying magnification. Navigation in the zoom-able surface and arrangement of a number of documents at different locations in a single view can be problematic for novice users. Pad++ relies only on spatial and visual attributes of information.

LifeStreams organizes documents by temporal attributes on a linear timeline.[15] Access to information can be done through either a time-ordered list of documents or a search tool. While temporal attributes of information are exploited to access information, spatial and visual attributes are totally neglected. Also, once documents are accessed and opened, the approach provides no additional layout organization facility besides what is provided by the window manager.

In LifeLines,[16] users can access documents from a compact temporal overview consisting of multiple time-lines each characterizing different aspects of the information through direct manipulation. Work on the Visage project[17] has introduced information-centric techniques in which individual data elements can be drag-and-dropped into different views to generate custom visualizations, such as tables, bar graphs, or geographial maps. Feiner describes a hypertext system which supports the creation of large graphical documents with an arbitrary directed graph structure, with graphical information hiding and structure manipulation capabilities.[18]

Lansdale observed that current interfaces do not support the variety of activities people use to organize and access information.[19] He observed that people remember far more about pieces of information than labels (nominal), such as when it was received (temporal), what it looks like (visual), where it was put (spatial), and many other things. It is also suggested that information be accessible based on partial knowledge of its attributes. Shneiderman demonstrated a number of interesting strategies to coordinate information existing in multiple windows.[20]

There is a growing interest in personal role management not only in the human-computer interaction area but in other areas as well. Researchers in the database area have extended the current object model with roles.[21]

An early study by Bury *et al*.[22] (1985) comparing user performance in windowed systems to non-windowed systems revealed that task-completion times in windowed systems can be longer due to window arrangement time. However, in a detailed analysis, the actual times spent on task execution were found to be shorter, and the error rates were significantly lower in windowed systems.

Bly and Rosenberg[23] compared user performance of tiled and overlapping window management strategies for regular and irregular tasks, where regularity is determined by the layout of information in a window. Their results favored tiled windows for regular tasks. For irregular tasks, however, expert performance was faster in overlapping windows, whereas novice performance was faster in tiled windows. Lane *et al*.[24] also compared tiled and arbitrary overlap strategies for multi-window searches. Their results also indicated faster novice user performance for the tiled strategy.

Gaylin[25] observed that the number of window operations that are used to switch the active window set constitutes 63 per cent of all the operations in an independent overlapped window manager. This result supports the findings by Bannon *et al*.[26]

that people switch among tasks frequently, forcing them to change the visible set of windows on the screen. According to Gaylin's observations, create and delete window operations accounted for about 15 per cent of total operations, whereas move and resize operations accounted for 6 per cent, with twice as many moves as the resizes.

Gaylin also measured window operation frequencies during log-on, as users set up their computers in a typical work configuration. Although the most frequently used commands are still those used to switch the active windows, window creation operations accounted for 17 per cent, move operation for 17 per cent, and resize for 12 per cent.

Gaylin used these window operation frequencies to create a windowing system benchmark. We believe that a more reliable benchmark test should be based on task domain operations, not interface domain operations.

Card *et al.*[27] introduced cost of knowledge characteristic function for characterizing information access from dynamic displays, where the cost is determined as the time to access information.

EXPERIMENTS

In our evaluation, we measured user performance on task environment setup, task environment switching, and task execution (Figure 10).

Task environment setup is the act of accessing information objects needed for the task, opening windows for them, and arranging the layout. An example would be for programmers to open source code modules in multiple windows and to arrange them on the screen.

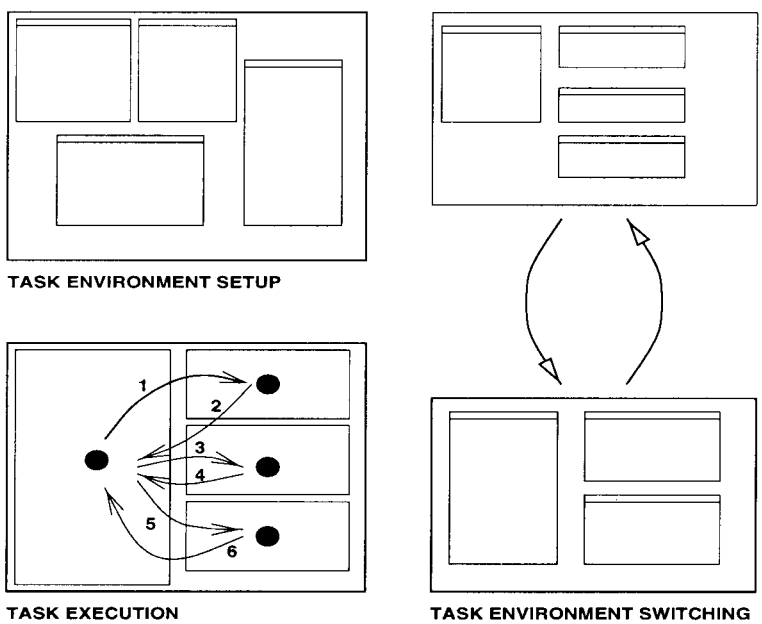

Figure 10. Three categories for window benchmarking: task environment setup, switching, and task execution

Task environment switching is the act of changing the screen contents to an existing environment setup. An example would be to switch to reading specifications in the middle of programming.

Task executions are actions with information contained in windows in a task environment layout. An example would be looking sequentially through many job descriptions to find the best paying job.

We identified four task execution types: Sequential scanning, comparison, determine context+scan, and recall context+scan (Figure 11).

Sequential scanning is looking sequentially through a number of information sources for a certain attribute of the information, such as the job salary. *Comparison* is comparing a number of information sources based on one or more attributes, such as job descriptions or benefits. It is different from sequential scanning because users tend to glance back and forth at information sources multiple times till they comprehend the distinctions well enough to make a judgment. *Determine context+scan* is a filtering based on an attribute to establish a context for further scanning. For example, once a decision is made to seek jobs in California, this context enables the users to limit scanning only to California jobs. In *recall context+scan*, the context is not determined, but rather recalled based on previous interaction with the same information sources.

We are aware that not all the tasks users do with computers are this regular and this list is not complete. We chose these four types of task execution because of their significance in personal role management.

Subjects

Twelve computer science graduate students with 11 of them having more than five years experience with window systems participated in the experiments. Seven out of 12 of the subjects had experience with three or more windowing systems and 9 out of 12 use a windowing system for more than 20 hours weekly.

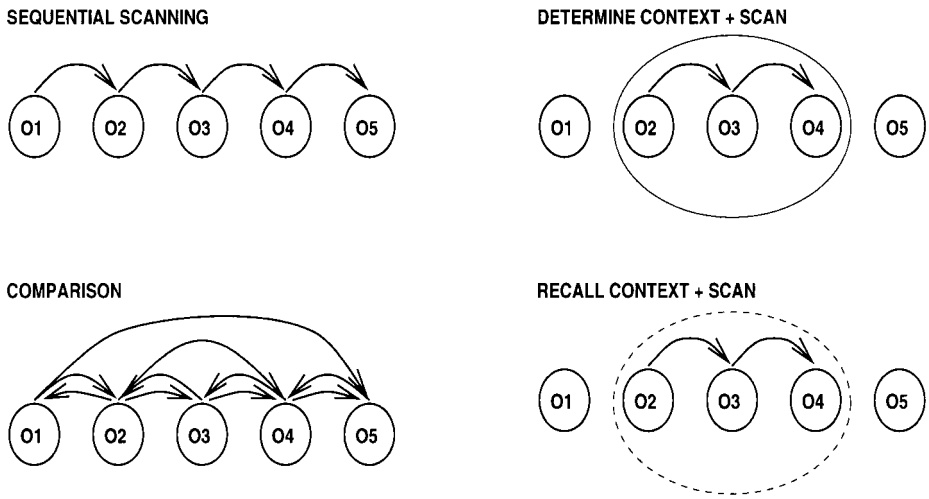

Figure 11. Task types: sequential scanning, comparison, determine context + scan, and recall context + scan

Design

The experiment design was a within-subject counterbalanced design with 12 subjects, with three subjects per group. Each subject was tested on both of the interfaces but the order of interfaces was counterbalanced for half of the users. To reduce the chance of performance improvement in the second interface, a parallel set of questions was used on the second interface. The order of the question set was also reversed for half of the subjects in each group. Since all four permutations ($E_1 \rightarrow I_2$, $E_2 \rightarrow I_1$,$I_1 \rightarrow E_2$, $I_2 \rightarrow E_1$) were included, results are presented for aggregated groups. The tasks in both of the question sets match each other in the same order. To verify the results obtained from the within-subject design, we extracted a between-subjects data from the initial study with reduced number of subjects (six subjects in two groups, only the first interface timings are counted). Also to investigate interface ordering effects and question set differences we ran three-way ANOVA (Interface order, Question set, Interface type) statistics.

Tasks

Each subject was tested on the information store of a hypothetical student. This student is enrolled in two courses this semester: Software Engineering and Computer Networks. Course materials and partners, homework, and correspondence with the professor, TAs, and classmates are organized in a hierarchical structure for each course. This student has a number of other roles like the organization of a birthday party, home duties, and job responsibilities. There are two projects at the job that the student is responsible for. Project materials, like the programming code, documents, reports, and correspondence with the partners and the boss are organized also in a hierarchical structure for each project.

User performance was measured on task environment setup, task environment switching, and task execution on three different degrees of task environment complexities: low, medium, and high, represented by 2, 6, and 12 windows, respectively. In the context of the student role, the task environments have been chosen as follows: low complexity task environment included two documents of a course project, medium complexity task environment included six e-mail messages from the boss at the job, and high complexity task environment included twelve pieces of project programming code at the job. Actual tasks for medium task environment complexity used in the experiment are:

(a) *Task environment setup:* Open all six e-mails from my Multimedia project boss Rich regarding new positions in the company.
(b) *Task execution (Sequential scanning):* Which of the positions require ‘TCP/IP programming experience of more than two years?’
(c) *Task execution (Comparison):* Which position has the longest list of requirements?
(d) *Task execution (Determine context+scan):* How many of the positions are in the Networking group?
(e) *Task execution (Recall context+scan):* What is the maximum wage for the positions in the Networking group?
(f) *Task environment switching:* Switch to the Analysis and Capability documents in the Software Engineering course.

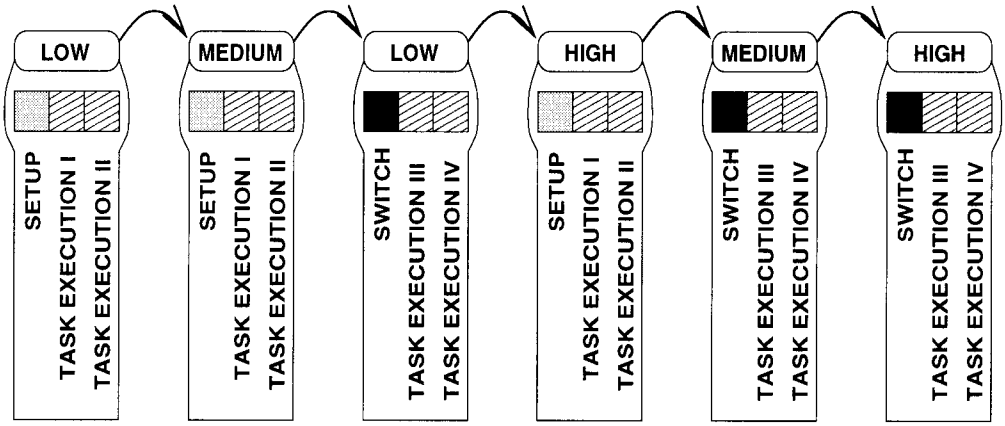

Figure 12. An example task schedule

An example task schedule for a subject is shown in Figure 12.

Each subject was given a similar six session schedule, with the order of task environment complexities varied. Two task executions are given at each session, a total of four for each task complexity, covering all task execution types. According to this schedule each subject makes three task environment setups at three different task environment complexities, three task environment switchings, and 12 task executions total.

Training

A 15-minute training session was given to each user for both of the interfaces, supplemented with a practice test. Users were expected to develop strategies for handling multiple windows in both of the interfaces during this practice. Users were also given a five-minute training session on the information hierarchy used in the experiment.

Training of the Elastic Windows interface began with the hierarchical coloring scheme, and the elastic nature of windows with the proportional space allocation

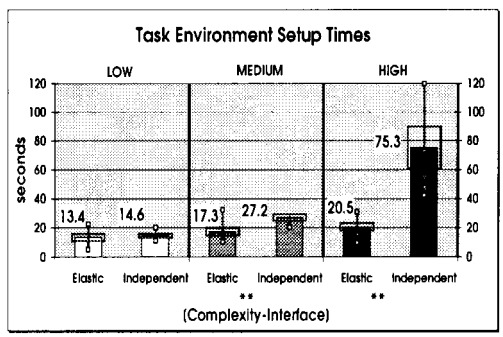

Figure 13. Average task environment setup times (for $p < 0.05$ ** for $p < 0.01$ indicating significance level)*

strategy. It covered opening/closing, resizing, packing/unpacking, and maximizing a hierarchy of windows.

Training of the independent windows interface covered a similar set of tasks, including opening a window, iconifying and reopening windows, resizing, and closing windows as well as traversing the information hierarchy using the file manager.

Hypothesis

Elastic Windows with multiple window operations yields faster performance than independent overlapped windows for expert users of independent overlapping windowing systems for task environment setup, switching, *and* task execution *for* medium *and* complex *task environments*.

Independent variables were the windowing interface (Elastic Windows and Independent Overlapping Windows), and task environment complexity (2, 6, and 12 windows). Dependent variables were task environment setup times, task environment switching times, and task execution times.

Task environment setup time is determined by the duration to bring up all windows related to the task. Task switching time is determined by the duration to switch the active window set. Task execution time is the duration, given a question about the information on multiple windows, it takes the subject to find the necessary information on windows and reply. It includes the necessary window arrangement time.

Procedure

The subjects were given a brief description of the experiment, filled out a subject information sheet, and signed a consent form. The experiment took about an hour, including the training and practice test. Subjects were free to ask any questions during the training session and before starting each task during the experiment.

The systems compared in the experiments are the Elastic Windows interface and a twm-clone window manager with the Open Windows file-manager both running on a Sun Sparc 20, using SunOS operating system under X windows.

RESULTS AND OBSERVATIONS

User performance

Task environment setup

Task environment setup times for Elastic Windows for medium and high complexity treatments were found to be less than that of the Independent Overlapping Windows with a statistical significance at the 0.01 level. For the high complexity situation, users performed about four times faster with Elastic Windows (20.5 seconds) than with Independent Overlapping Windows (75.3 seconds). The standard deviation for the high complexity task environment setup in the Independent Overlapping Windows is high due to the diverse approaches taken by the subjects in the organization of windows. Statistics for the between-subjects design confirmed these

results. The ANOVA statistics found no significant difference related to interface ordering and question set differences.

In Elastic Windows, the steps for setting up a task environment include opening a container window, selecting multiple task related objects, and dragging and dropping them in the container window. It might be necessary to maximize the container window to full screen for better visibility. Average task environment setup times stayed nearly constant at all task complexities as shown in Figure 13. Standard deviations are shown as rectangles over the bars in the chart, and the minimum and maximum times are shown as a vertical line.

In the Independent Overlapping Windows, each icon has to be double-clicked and the windows placed appropriately on the screen, one by one. In this approach, the setup times are heavily dependent on the number of windows. However, the dependency is more than linear since as the number of windows on the screen increases, it becomes much more difficult to arrange windows.

Multiple selection and open can easily be added to the existing windowing systems, but what is lacking is the framework to identify and operate on multiple windows as a group.

Task switching

All task switching times were found to be shorter with the Elastic Windows interface with a statistical significance level of 0.05, except for low to medium and low to high complexity environment switchings. The between-subject analysis, however, found low to high switching statistically significant since an outlier was detected using the Grubbs test. The medium to low switching turned out to be not statistically significant. ANOVA analysis was not made for task switching analysis since the number of subjects (3) were low in each treatment.

The average task switching times are shown in Figure 14.

Diverse strategies in switching among environments, led to high variances in performance times. Still, the average time to do a task environment switch was nearly constant (4.8 to 7.3 seconds) independent of the environment complexity. In the Independent Overlapping Windows, however, the switching time increased as task environment complexity increased (12.1 to 45.2 seconds). This is mainly due to the one window at a time approach. Providing an overview and a set of

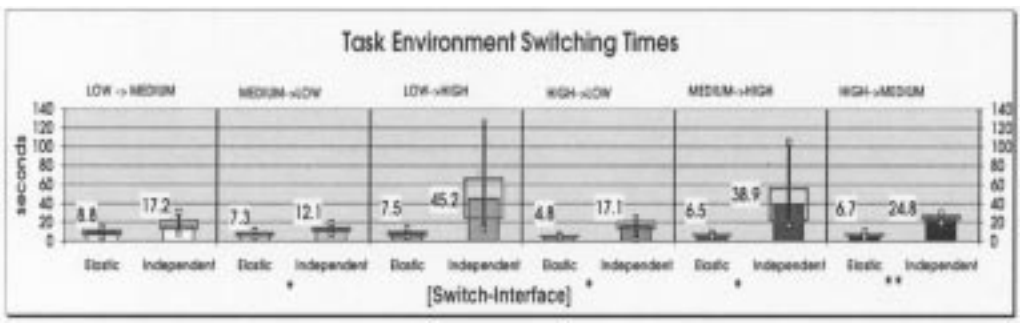

Figure 14. Average task environment switching times (for $p < 0.05$ ** for $p < 0.01$ indicating significance level)*

workspaces, as in Rooms,[7] would certainly make task switching time independent of the number of windows involved, but Rooms offers only two levels.

Task execution

Task execution times for all task complexities and task execution types were statistically significantly shorter for Elastic Windows at the 0.05 level, except for sequential scanning and recall context+scan for the low complexity treatment. The between-subjects analysis confirmed these results. ANOVA statistics found no statistically significant ordering effect, or question set differences except for sequential scanning and recall context+scan for the medium complexity treatment. In sequential scanning an interaction is found between interface type and question set. In recall context+scan an interaction is found in the combination.

In sequential scanning, having a stable layout during the task execution helped subjects greatly. In Elastic Windows, windows are well-organized, side by side, and during task execution subjects did not find it necessary to manipulate (resize, move) windows. However, in Independent Overlapping Windows, the layout was continuously changing, windows were raised, moved, and resized frequently, due to limited screen space. Subjects produced dramatic changes from the initial layout during task execution. These disruptive changes were more prevalent as task environment complexity increased.

In Comparison, having windows side by side in Elastic Windows helped users to compare window contents. Since windows are well organized, users adopted a visual approach in comparing window contents, and eliminated some windows immediately. However, in Independent Overlapping Windows, users had to look at each window one by one, changing the layout constantly, which made it harder to do the comparison after a while. The situation was more severe in the high complexity environment (Figure 15). Users performed a comparison task in 10.9 seconds with Elastic Windows and in 135.4 seconds with Independent Overlapping Windows, thereby achieving more than ten-fold performance speed-up.

In determine context+scan, subjects using Elastic Windows maximized a subset of the windows belonging to the context, enabling them to focus on the context more easily due to larger screen space allocated. In Independent Overlapping Windows, however, subjects did not reorganize the layout but relied on the scrollbar.

Recall was easier in the Elastic Windows interface because of the more stable window organization across task executions. Subjects stated that it was easier to remember window locations than in the Independent Overlapping Windows. Since the window organization was modified in the overlapping windows interface for each task execution in the sequence, the locational memory of users was lost. Task execution times were statistically significantly shorter with Elastic Windows for medium and high complexities, at the 0.05 level. In the low complexity task environment with only two windows on the screen, it was not difficult to recall window locations.

Some of the subjects had one task execution with comparably low performance in the Elastic Windows interface, where subjects had hard times thinking of a strategy for the task. This is mostly due to the relative inexperience of the subjects with the Elastic Windows interface as opposed to their lengthy experience with the popular Independent Overlapping Windows interfaces.

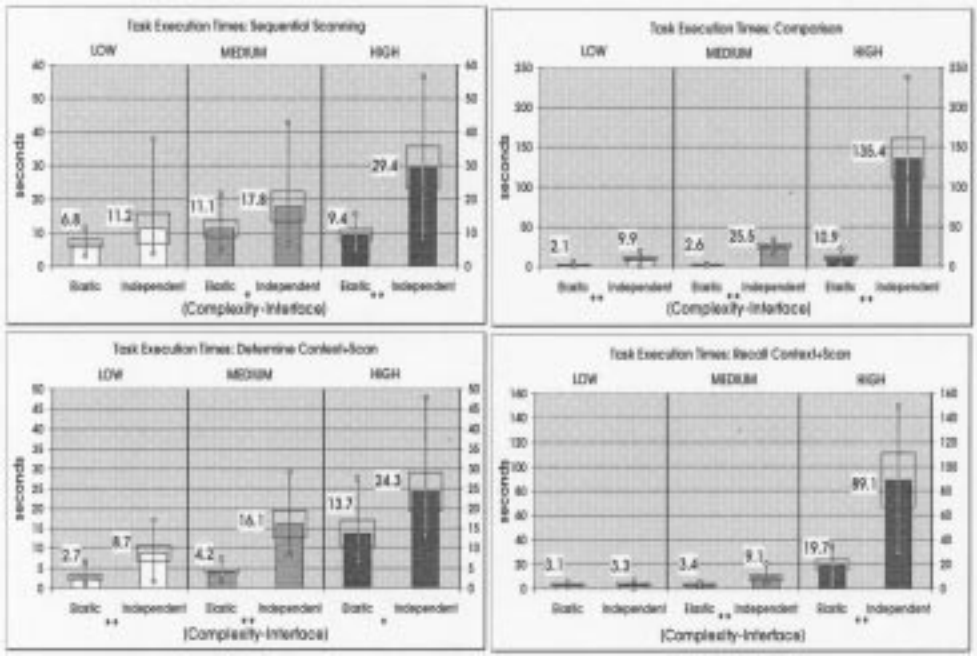

Figure 15. Average task execution times (for $p < 0.05$ ** for $p < 0.01$ indicating significance level)*

In summary, multi-window operation facilitated by the hierarchical organization of windows proved to be faster for task environnment setup, switching, and task executions than single window operations in current approaches. These results suggest promising possibilities for multiple window operations and hierarchical nesting, which can be applied to the next generation of tiled as well as overlapped window managers. They should enable users to more readily deal with increasingly complex tasks. Especially, tasks with multiple windows (more than two) are likely to benefit from multi-window operations.

Subject interviews

After the experiments subjects were debriefed about their usual use of multiple windows. Most of the subjects expressed a preference to use more windows for some tasks, given efficient means to do so. They described opening multiple copies of the same source file to view different parts of the code, thereby avoiding disruptive scrolling and find commands.

Some subjects said that, although it was not easy to see the hierarchy at first, they claimed to get used to it after several tasks. According to our observations during the experiment, subjects were initially following the hierarchy to access information; however, after some time, they started to use their spatial memory and access information directly based on that knowledge. This observation was confirmed by most of the subjects. Some subjects, however, had no problems visualizing the hierarchy. One subject said that he liked the overview of hierarchical roles as a guide to his daily tasks.

Most of the users indicated preference for clicking to see contents. This could be due to their past practice on independent overlapping windows systems, but also their desire for a quick look at window contents. Subjects made a number of suggestions for the Elastic Windows interface about this and other features.

CONCLUSION

We believe that there is an opportunity to improve today's window management strategies. This paper suggests requirements for future windowing systems, and then describes the Elastic Windows approach in detail. Its hierarchical structure of window organization enables users to do multiple window operations by applying window operations on groups of windows. Our experiment compared Elastic Windows with Independent Overlapping Windows in terms of user performance times on task environment setup, switching, and four task execution types. We found statistically significant performance differences in favor of the Elastic Windows interface for most of the tasks. A ten-fold performance speed-up was achieved for a comparison task at the high complexity situation. We are working on extending and formalizing our evaluation method, possibly leading to a window benchmarking test based on task domain actions.

Current implementation is based on the space-filling tiled layout strategy. In tiled approaches window sizes may not always conform to window contents. However, the ideas of hierarchical window organization and multiple window operations are applicable to other layout strategies.

These results suggest promising possibilities for multiple window operations and hierarchical nesting, which can be applied to the next generation of tiled as well as overlapped window managers. They should enable users to more readily deal with increasingly complex tasks. However, our experiments measured only expert user performance. While the Elastic Windows interface offers more powerful window management facilities, it might require more user training. Studies on novice users with extended set of tasks and long-term usage will provide important information on its acceptance for a larger population.

Role management was not explicitly tested in this study, but users appeared to grasp this novel layout strategy and use it competently. A future study will focus on the benefits of role management and alternate layouts to support it.

Acknowledgements

We are grateful to Kent L. Norman for his contribution in the analysis of experiment results. Special thanks go to Visix Software Inc. for their donation of the Galaxy Application Environment used in the development of Elastic Windows. This material is based upon work supported by the National Science Foundation under Grant No. NSF IRI 96-15534, and by IBM.

REFERENCES

1. B. Johnson and B. Shneiderman, 'Space-filling approach to the visualization of hierarchical information structures', *Proc. IEEE Visualization '91*, 1991, pp. 284–291.
2. S. K. Card, M. Pavel and J. E. Farrell, 'Window-based computer dialogues', *Proc. INTERACT '84, First IFIP Conference on Human-Computer Interaction*, London, UK, 1984, pp. 355–359.
3. M. J. Kahn and E. Charnock, 'How to prevent "windowitis" in your graphical interface?', *Proc. Silicon Valley Ergonomics Conference & Exposition, ErgoCon'95*, 1995, pp. 18–25.

4. B. Shneiderman and C. Plaisant, 'The future of graphic user interfaces: Personal role managers', *People and Computers IX*, Cambridge, UK, 1994, pp. 3–8. Cambridge University Press.
5. C. Plaisant and B.Shneiderman, 'Organization overviews and role management: Inspiration for future desk-top environments', *Proc. IEEE 4th Workshop in Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1995, pp. 14–22.
6. E. Kandogan and B. Shneiderman, 'Elastic Windows: Improved spatial layout and rapid multiple window operations', *Proc. Advanced Visual Interfaces '96*, New York, 1996, pp. 29–38. ACM.
7. A. Henderson and S. K. Card, 'Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface', *ACM Trans. Graphics*, **5**(3), 211–243 (1986).
8. T. W. Malone, 'How do people organize their desks? Implications for the design of office automation systems', *ACM Trans. Office Information System*, **1**(1), 99–112 (1983).
9. E. S. Cohen, E. T. Smith and L. A. Iverson, 'Constraint-based tiled windows', *IEEE Computer Graphics and Applications*, **6**(5), 2–11 (1986).
10. W. Teitelman, 'A tour through cedar', *IEEE Software*, **1**(2), 44–73 (1984).
11. B. Myers, 'Window interfaces: A taxonomy of window manager user interfaces', *IEEE Computer Graphics and Applications*, **8**(5), 65–84 (1988).
12. J. Dumas and P. Parsons, 'Discovering the way programmers think about new programming environments', *Communications of the ACM*, **38**(6), 45–56 (1995).
13. S. Card, G. Robertson and W. York, 'The webbook and the web forager: An information workspace for the world-wide web', *Proc. ACM CHI '96 Conference–Human Factors in Computing Systems*, 1996, pp. 111–117.
14. B. B. Bederson and J. D. Hollan, 'Pad++: A zooming graphical interface for exploring alternate interface physics', *Proc. ACM UIST'94, User Interface Software and Technology Conference*, 1994, pp. 17–26.
15. E. Freeman and D. Gelernter, 'Lifestreams: A storage model for personal data', *ACM SIGMOD Bulletin*, **25**(1), 80–86 (1996).
16. C. Plaisant, B. Milash, A. Rose, S. Widoff and B. Shneiderman, 'Lifelines: Visualizing personal histories', *Proc. ACM CHI-96 Conference–Human Factors in Computing Systems*, 1996, pp. 221–227.
17. S. F. Roth, P. Lucas, J. A. Senn, C. C. Gomberg, M. B. Burks, P. J. Stroffolino, J. A. Kolojejchick and C. Dunmire, 'Visage: A user interface environment for exploring information', *Proc. IEEE Symposium on Information Visualization*, 1996, pp. 3–12.
18. S. Feiner, 'Seeing the forest for the trees: Hierarchical display for hypertext structure', *Proc. ACM UIST'90, User Interface Software and Technology*, 1990, pp. 205–212.
19. M. Landsdale, 'The psychology of personal information management', *Applied Ergonomics*, **19** 55–67 (1988).
20. B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction: Second Edition*, Addison Wesley, Reading, MA, 1992.
21. G. Gottlob, M. Schreftl and B. Roeck, 'Extending object-oriented systems with roles', *ACM Trans. Information Systems*, **14**(13), 268–296 (1996).
22. K. F. Bury, S. E. Davies and M.. J. Darnell, 'Window Management: A review of issues and some results from user testing', *Technical Report HFC-53*, IBM Human Factors Center, San Jose, CA, June 1985.
23. S. Bly and J. Rosenberg, 'A comparison of tiled and overlapping windows', *Proc. ACM CHI'86 Conference–Human Factors in Computing Systems*, 1986, pp. 101–106.
24. J. C. Lane, S. P. Kuester and B. Shneiderman, 'User interfaces for a complex robotic task: A comparison of tiled vs. overlapped windows', *Technical Report January*, University of Maryland, Computer Science Department College Park, MD, CS-TR-3784, 1997.
25. K. B. Gaylin, 'How are windows used? Some notes on creating empirically-based windowing benchmark task', *Proc. ACM CHI'86 Conference–Human Factors in Computing Systems*, 1986, pp. 96–100.
26. L. Bannon, A. Cypher, S. Greenspan and M. L. Monty, 'Evaluation and analysis of users' activity organization', *Proc. ACM CHI'83, Human Factors in Computing Systems Conference*, 1983, pp. 54–57.
27. S. K. Card, P. Pirolli and J. D. Mackinlay, 'The cost-of-knowledge characteristic function: Display evaluation for direct-walk dynamic information visualizations', *Proc. ACM CHI'94 Conference–Human Factors in Computing Systems*, 1994, pp. 238–244.