Elastic Windows: Evaluation of Multi-Window Operations

Eser Kandogan and Ben Shneiderman*
Department of Computer Science,
Human-Computer Interaction Laboratory
*Institute for Systems Research
University of Maryland, College Park, MD 20742
kandogan@cs.umd.edu, ben@cs.umd.edu

ABSTRACT
Most windowing systems follow the independent overlapping windows approach, which emerged as an answer to the needs of the 1980s' technology. Due to advances in computers and display technology, and increased information needs, modern users demand more functionality from window management systems. We proposed Elastic Windows with improved spatial layout and rapid multi-window operations as an alternative to current window management strategies for efficient personal role management [12]. In this approach, multi-window operations are achieved by issuing operations on window groups hierarchically organized in a space-filling tiled layout. This paper describes the Elastic Windows interface briefly and then presents a study comparing user performance with Elastic Windows and traditional window management techniques for 2, 6, and 12 window situations. Elastic Windows users had statistically significantly faster performance for all 6 and 12 window situations, for task environment setup, task environment switching, and task execution. For some tasks there was a ten-fold speed-up in performance. 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.

Keywords
Window Management, Multi-window operations, Personal Role Management, Tiled Layout, User Interfaces, Information Access and Organization.

INTRODUCTION
As Card et al. [5] stated, an analysis of window management strategies can only be done by a careful consideration of the tasks for which windows are used. They attempted to categorize tasks by the functions provided by windows which they listed as:

- More information
- Access to multiple sources of information
- Combining multiple sources of information
- Independent control of multiple programs
- Reminding
- Command context/active forms
- Multiple representations

Most current windowing systems follow the independent overlapping windows approach, which emerged as an answer to the needs of 1980's applications and technology. These windowing systems no longer provide efficient means to serve the functions in this list for today's information-intensive applications. With advances in computer networks, especially the Internet, users are collecting more information in image, video, sound, and structured text formats.

With the introduction of windows, users can employ spatial attributes like location of icons or open windows to access information. However, current systems provide limited capabilities of icon and window organization; generally a single screen space where icons and windows can be placed independently anywhere on the screen. As a result, the computer screen becomes cluttered and windows are hidden, making it harder to access information using spatial attributes.

Access to and use of multiple sources of information or multiple representations are difficult because operations are performed one window at a time. Providing multiple window operations with a single action is likely to help users.

Novel approaches emphasize a docu-centric approach (Microsoft OLE and Apple's OpenDoc) in which documents become more important and applications fade into the background. The enriched document can contain various types of objects such as text, image, video, sounds, spreadsheets, etc.

Although these innovations are one step toward achieving a computer working environment in harmony with users' perceptions of their work, an effective organization of information according to users' roles that reflects this perception may bring further benefits [19, 16].

The key to personal role management is organizing information according to the roles of an individual. When users are working in a role, they have the most relevant objects regarding that role like schedules, documents, tools, 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 and abandon roles as well as extend and modify the role hierarchy.

Figure 1: Hierarchical Organization of a Professor's Roles: University Research and Teaching, Industry, and Personal

Our earlier work [12] stated the requirements for future windowing systems. A more complete list is as follows:

- Support a unified framework for information organization and coordination according to users' *roles*.
- Provide a visual, spatial layout that matches *semantics*.
- Support *multi-window* operations for fast arrangement of information.
- Support information access with *partial knowledge* of its *nominal, spatial, temporal,* and *visual* attributes and relationships to other pieces of information.
- Allow fast *switching* and resumption among roles.
- Free users' cognitive resources to work on *task domain operations* rather than computer domain operations.
- Use *screen space efficiently* and productively for tasks.

The next section gives a brief description of the Elastic Windows approach, followed by an analytic comparison of windowing systems. Next, the study comparing performance of Elastic Windows to traditional Independent Overlapping Windows is described in detail, along with the results and observations made.

ELASTIC WINDOWS

The 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. Hierarchical grouping of windows is indicated by gradually changing border colors according to the level of the window (Figure 1).

Figure 1 displays the hierarchical organization of different roles of a university professor. This professor is advisor to a number of graduate students in a number of research projects, teaches two courses this semester at the university, 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.

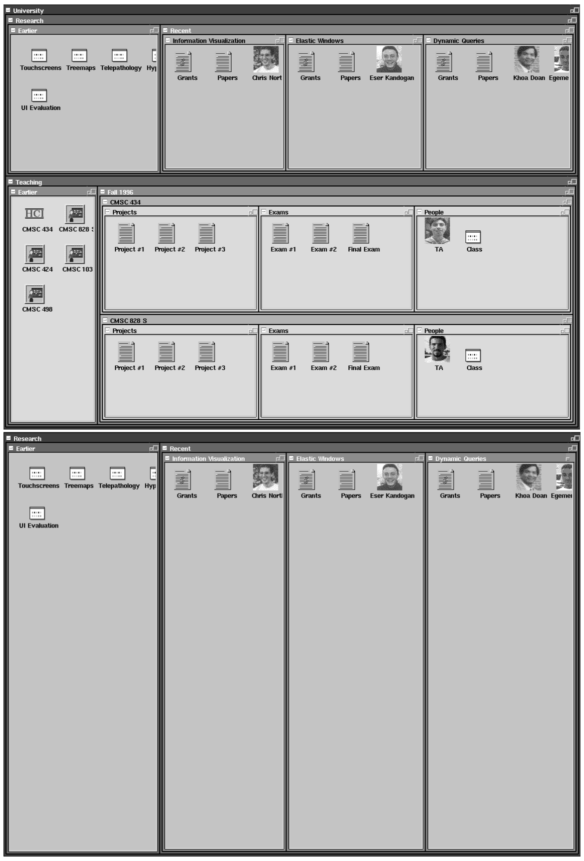

Figure 2: a) Maximize university roles from Figure 1 b) Maximize research roles within the university

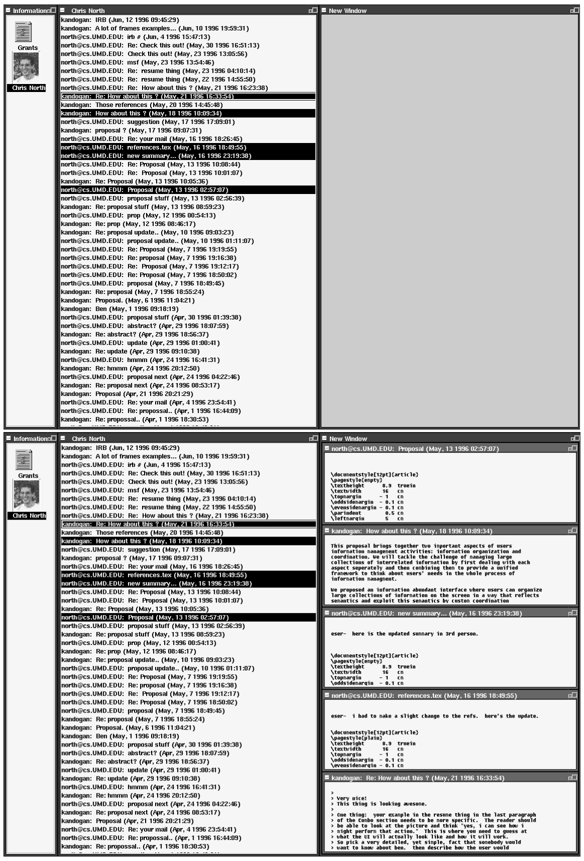

Figure 3: a) An empty container window opened on the right b) Multi-window open for selected items

Hierarchical grouping provides role-based context for information organization. It also supports graphical information hiding capability where window hierarchies can be collapsed into a single icon (or other primitives) making the approach scalable. Collapsed hierarchy of windows can be saved and retrieved, which allows users to reuse a previous window organization. Besides, layouts can be saved under different names giving users flexibility in using alternative layouts for different subtasks within the same context.

Current window management strategies have a limited notion of workspace. Most of the systems provide only one screen, whereas more novel systems, following the Rooms approach [11], provide multiple virtual screen spaces where windows can be placed in any of these spaces. Rooms also provides an overview where users can look at thumbnail images of the screen layouts and use the overview to switch to these screens. Users are limited to an overview level and the workspace level. In Elastic Windows, however, multi-level task focus is provided by allowing users to make any window full screen at any point in the hierarchy (Figure 2).

Multiple Window Operations

Typically, people organize papers on their desk as piles, and move all of them simultaneously. Malone [14] found that users like to group items spatially. 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 groups of windows at any level of the hierarchy. The results of operations are propagated to lower level windows inside that group recursively. In this way, a hierarchy of windows can be packed, resized, or closed with a single operation. Operations like multi-window open, close, resize, maximize, and pack enable users to change the window organization quickly to compare, filter, and apply the information.

In Elastic Windows, window groups can be created by opening a container window and dragging and dropping selected items inside this window (Figure 3). Separate windows are opened for each item in the selection as a member of the group surrounded by the container-window borders. Multiple items can be added to or removed from the group at any time. It is also possible to open a new container window within another container window to create hierarchical windows.

Space-filling Tiled Layout

We took a space-filling tiled approach 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.

In Elastic Windows, groups of windows stretch like an elastic material as they are resized, and other windows shrink proportionally to make space (Figure 4). Users are given flexibility in the placement of sub-windows in a group. There is no strict horizontal or vertical placement rule within window groups. The extent of window operations is limited to the windows in the same group and their sub-windows. Effects in the upper levels are propagated down to sub-windows recursively.

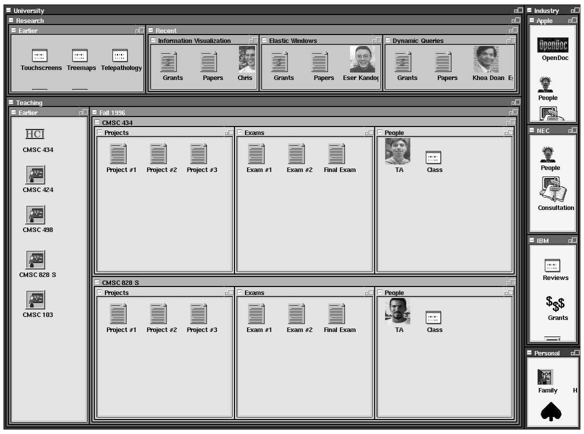

Figure 4: Elastic resizing of Teaching window on the original layout of Figure 1.

EVALUATING WINDOWING SYSTEMS

A 1985 study by Bury et al. [4] 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 [3] compared user performance of tiled and overlapping window strategies for regular and irregular tasks, where regularity is determined by the organization of information in a window. Their results supported tiled windows for regular tasks. For irregular tasks, however, expert performance was faster in overlapping windows, whereas novice performance was faster in tiled windows.

Gaylin [9] observed that the number of window operations to switch the active window set constitutes 63% of all the operations in an independent overlapped window manager. This result supports the findings by Bannon et al. [1] 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%, whereas move and resize for 6%, with twice as many moves as 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%, move operation for 17%, and resize for 12%.

Gaylin used window operation frequencies to create a windowing system benchmark. We believe that a more reliable benchmark test should be based on task-domain rather than interface-domain operations (e.g. window operations).

In our evaluation, we measured user performance on *Task Environment Setup, Task Environment Switch,* and *Task Execution* (Figure 5).

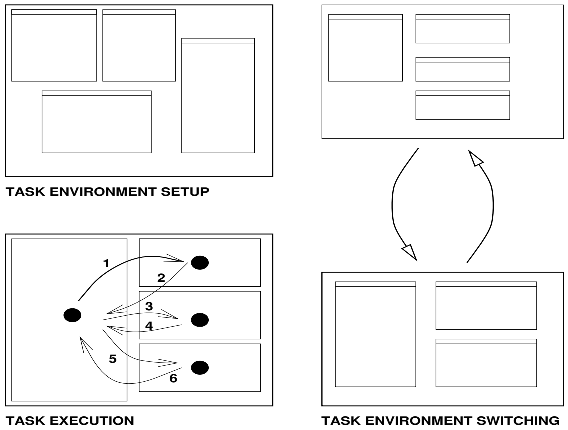

Figure 5: Three categories for evaluation.

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.

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 6).

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 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 to only California jobs. In Recall Context+Scan, the context is not determined rather recalled based on previous interaction with the same information sources. It is designed to test how well the windowing system supports recall based on spatial attributes.

We are aware that not all the tasks users do with computers are this regular and this list is not complete. We have chosen these four types of task execution because of their significance in personal role management. Task executions types chosen cover basic information management tasks such as a quick

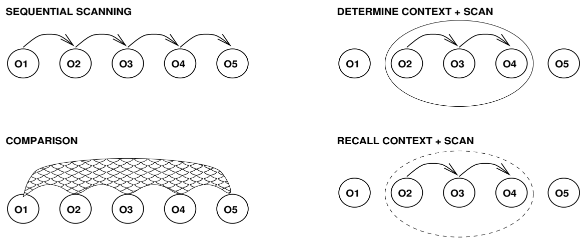

Figure 6: Task execution types

scan of multiple information to gain an overview, comparison of multiple information, and filtering of a set of information from a larger set.

EXPERIMENTS

Subjects

Twelve computer science graduate students, 11 of which had more than 5 years experience with windowing systems, participated in the experiments. Seven subjects had experience with 3 or more windowing systems and 9 used a windowing system for more than 20 hours weekly.

Design

The experiment design was a within-subject counterbalanced design with 12 subjects. Each subject was tested on both of the interfaces but the order of interfaces was reversed for half of the users. To reduce the chance of performance improvement, 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. The order of the tasks in both of the sets was the same. Since all four permutations were included, results are presented for aggregated groups. Paired T-tests were used to compare user performance at the 0.05 level of statistical significance.

Hypothesis

Elastic Windows with multiple window operations yields faster performance than independent overlapped windows for expert users of current 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, switching times, and task execution times.

Tasks

Subjects were tested using the information hierarchy of a hypothetical student. User performance was measured on task environment setup, task environment switching, and task execution at all three task complexities. In the student role context, the task environment complexity was: Low (2 reports of a course project), medium (6 e-mail messages from the boss), and high (12 modules of programming code).

Each subject performed three task environment setups, one from each complexity, three task environment switchings, and a total of 12 task executions, covering all task execution types at all three complexities. The order of task environment complexities were varied across subjects to test all combinations of task environment switchings.

Training

Prior to the experiment, each subject was given 15 minutes of training 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 5 minutes of training on the information hierarchy used in the experiment.

Training on the Elastic Windows interface began with the hierarchical coloring scheme, and the elastic nature of windows with the proportional space allocation strategy. It covered opening/closing, resizing, packing/unpacking, and maximizing a hierarchy of windows.

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

Procedure

The subjects received 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.

Systems

The Elastic Windows interface and a twm-clone window manager with the OpenWindows file-manager were both running on a Sun Sparc 20, using SunOS operating system under X windows.

RESULTS AND DISCUSSION

User Performance

Task Environment Setup Although average task environment setup time for Elastic Windows in the low complexity treatment happened to be less than that of the Independent Overlapping Windows, there was no statistically significant difference. For medium and high complexities, however, Elastic Window's setup times were lower, and the difference was statistically significant (Figure 7). Standard deviations are shown as rectangles over the bars in the chart, and the minimum and maximum times are shown as a vertical line. (* for 0.05, and ** for 0.01 level of statistical significance)

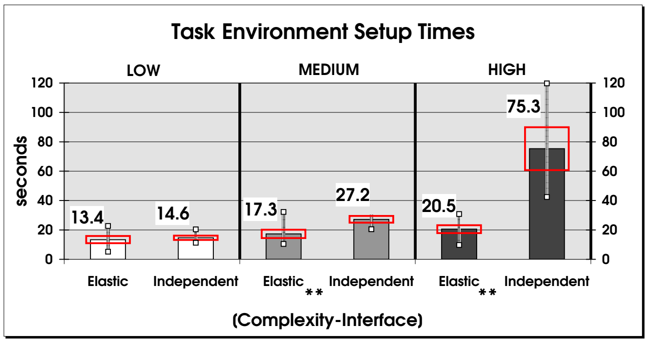

Figure 7: Task Environment Setup Times

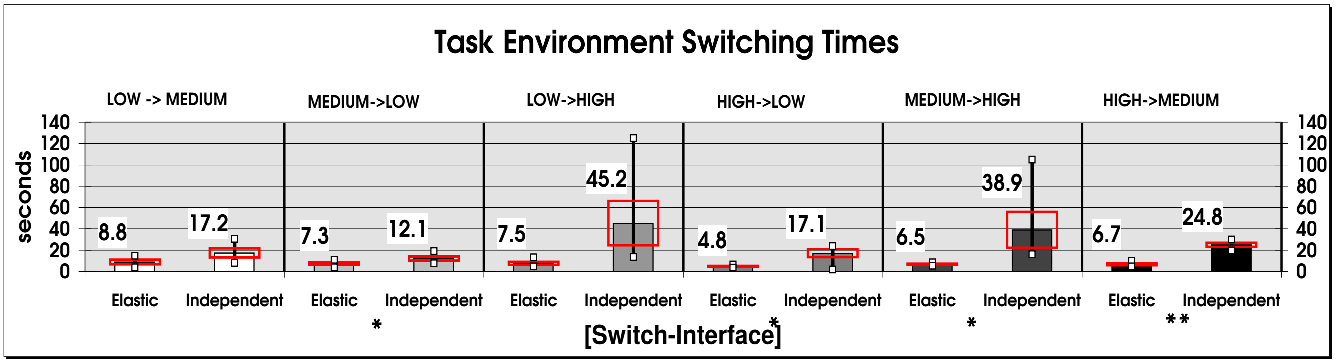

Figure 8: Task Environment Switching Times

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. Some subjects maximized the container window to full screen for more efficient utilization of the screen space. Average task environment setup times stayed nearly constant for all task complexities as shown in Figure 7. The increase was mostly due to the loading of larger number of files.

In Independent Overlapping Windows, each icon has to be double-clicked and the windows placed appropriately on the screen, one by one. 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. We believe that the high standard deviation for the high complexity task environment setup is due to the diverse approaches taken by the subjects in their organization of 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 Environment Switching All results supported the Elastic Windows interface. The differences were statistically significant except for low to medium and low to high environment switchings (Figure 8).

Elastic Windows allows multiple levels of workspaces where a hierarchy of windows at any level can be made to fill the whole screen. During the experiments, some of the subjects used three actions to enlarge a window to maximize, whereas some used only two. Variation among task switch performances was mainly caused by the number of actions to achieve maximization. Although the switching times from low to medium and low to high complexities were less for the Elastic Windows interface, the variation among the subjects prevented a possible statistically significant difference.

Diverse strategies in switching among environments, led to variances in performance times. Still, the average time to do a task environment switch was nearly constant, independent of the environment complexity. In the Independent Overlapping Windows, however, the switching time increased as task environment complexity increased. This was mainly due to the one window at a time approach. Providing an overview and a set of workspaces, as in Rooms [11], 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, except for Sequential Scanning and Recall Context+Scan for the low complexity treatment (Figure 9).

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 problem was more severe in the high complexity treatment.

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.

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. In the low complexity task environment with only two windows on the screen, it was not difficult to recall window locations.

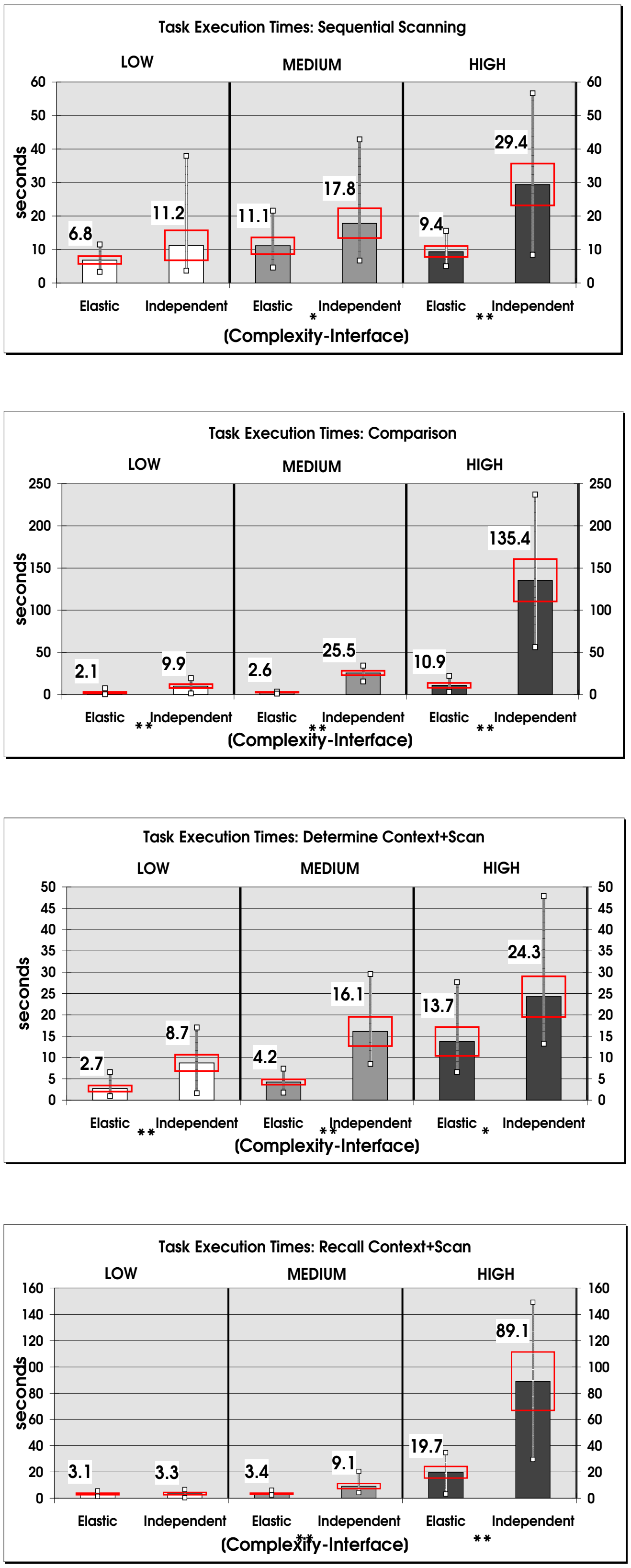

Figure 9: Task Execution Times

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 program code, thereby avoiding disruptive scrolling and find commands.

Some subjects claimed that, although it was not easy to see the hierarchy at first, they got 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 locational 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.

RELATED WORK

The Rooms system [11] 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. Rooms has no support for multiple window operations.

Recent research in more advanced information management user interfaces has generated a handful of interesting innovations. The WebBook work at Xerox extends the 2D desktop metaphor to a 3D office metaphor [6]. Pad++ introduced a novel technique for spatially organizing information on an infinitely zoom-able surface [2]. LifeStreams organizes documents by temporal attributes on a linear timeline [10]. In LifeLines [17], users can access documents from a compact temporal overview consisting of multiple time-lines each characterizing different aspects of the information through direct manipulation. IGD is 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 [8]. The Dylan programming environment uses a pane-based window system [7], which allows both horizontal and vertical panes, with a mechanism to create links between panes.

Lansdale [13] argues that people employ a number of different strategies to access information during their daily practices, and it would be beneficial to support those strategies in computer environments. In [18], a number of interesting strategies are introduced to coordinate information in multiple windows. Myers has an excellent taxonomy of early windowing systems [15].

SUMMARY AND FUTURE WORK

We believe that there is an opportunity to improve today's window management strategies. This paper suggest requirements for future windowing systems, and then reviews the Elastic Windows approach. 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 support of the Elastic Windows interface for most of the tasks. For some tasks there was a ten-fold speed-up in performance. We are working on extending and formalizing our evaluation method, possibly leading to a window benchmarking test based on task domain actions.

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.

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.

ACKNOWLEDGEMENT

We are grateful to Kent L. Norman for his contribution in the analysis of experiment results, Catherine Plaisant, Egemen Tanin, and Chris North for their reviews on the draft of this paper. Special thanks go to Visix Software Inc. for their donation of the Galaxy Application Environment used in the development of Elastic Windows. This research is supported by grants from the National Science Foundation under Grant No. NSF EEC 94-02384 and NSF IRI 96-15534, and by IBM.

REFERENCES

1. Bannon, L., Cypher, A., Greenspan, S., Monty, M. L., Evaluation and analysis of users' activity organization, *Proc. CHI'83, Human Factors in Computing Systems Conference*, ACM, New York, NY, (1983), pp. 54-57.

2. Bederson, B., B., Hollan, J., D., Pad++: A zooming graphical interface for exploring alternate interface physics, *Proc. UIST'94, User Interface Software and Technology Conference*, (1994), pp. 17-26.

3. Bly, S., Rosenberg, J., A comparison of tiled and overlapping windows, *Proc. CHI '86 Conference - Human Factors in Computing Systems*, ACM, New York, NY, (1986), pp. 101-106.

4. Bury, K. F., Davies, S. E., and Darnell, M. J., Window management: A review of issues and some results from user testing, *IBM Human Factors Center Report HFC-53*, San Jose, CA, (June 1985), 36 pages.

5. Card, S. K., Pavel, M., and Farrell, J. E., Window-based computer dialogues, *INTERACT '84, First IFIP Conference on Human-Computer Interaction*, London, UK, (1984), pp. 355-359.

6. Card, S., Robertson, G., York, W., The WebBook and the Web Forager: An information workspace for the World-Wide Web, *Proc. CHI'96 Conference - Human Factors in Computing Systems*, New York, NY, (1996), pp. 111-117.

7. Dumas, J., Parsons, P., Discovering the way programmers think about new programming environments, *Communications of the ACM 38*, 6, (June 1995), pp. 45-56.

8. Feiner, S., Seeing the forest for the trees: Hierarchical display for hypertext structure, *Proc. UIST'90, User Interface Software and Technology*, (1990), pp. 205-212.

9. Gaylin, K., B., How are windows used ? Some notes on creating empirically-based windowing benchmark task, *Proc. CHI '86 Conference - Human Factors in Computing Systems*, ACM, New York, NY, (1986), pp. 96-100.

10. Freeman, E., Gelernter., D., LifeStreams: A storage model for personal data, *ACM SIGMOD Bulletin 25*, 1, (March 1996), pp. 80-86.

11. Henderson, A., Card, S. K., Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface, *ACM Transactions on Graphics 5*, 3, (1986), pp. 211-243.

12. Kandogan, E., Shneiderman, B., Elastic Windows: Improved Spatial Layout and Rapid Multiple Window Operations, *Proc. Advanced Visual Interfaces '96*, ACM, New York, NY, (May 1996), pp. 29-38.

13. Lansdale, M., The psychology of personal information management, *Applied Ergonomics*, (March 1988), pp. 55-67.

14. Malone, T. W., How do people organize their desks? Implications for the design of office automation systems, *ACM Transactions on Office Information Systems 1*, pp. 99-112.

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

16. Plaisant, C., Shneiderman, B., Organization overviews and role management: Inspiration for future desktop environments, *Proc. IEEE 4th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, (April 1995), pp. 14-22.

17. Plaisant, C., Milash, B., Rose, A., Widoff, S., Shneiderman, B., LifeLines: Visualizing personal histories, *Proc. CHI '96 Conference - Human Factors in Computing Systems*, New York, NY, (1996), pp. 221-227.

18. Shneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction: Second Edition*, Addison Wesley Publ. Co., Reading, MA, (1992), Ch.9.

19. Shneiderman, B., Plaisant, C., The future of graphic user interfaces: Personal role managers, *People and Computers IX*, Cambridge University Press, (Aug 1994), pp. 3-8.