Taskposé: Exploring Fluid Boundaries in an Associative Window Visualization

Michael Bernstein
MIT CSAIL
Cambridge, MA
msbernst@mit.edu

Jeff Shrager
Symbolic Systems Program
Stanford University
Stanford, CA
jshrager@stanford.edu

Terry Winograd
Stanford HCI Group
Stanford, CA
winograd@cs.stanford.edu

ABSTRACT
Window management research has aimed to leverage users' tasks to organize the growing number of open windows in a useful manner. This research has largely assumed task classifications to be binary — either a window is in a task, or not — and context-independent. We suggest that the continual evolution of tasks can invalidate this approach and instead propose a fuzzy *association* model in which windows are related to one another by varying degrees. Task groupings are an emergent property of our approach. To support the association model, we introduce the WindowRank algorithm and its use in determining window association. We then describe Taskposé, a prototype window switch visualization embodying these ideas, and report on a week-long user study of the system.

Author Keywords
Task management, window management.

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—graphical user interfaces, windowing systems.

INTRODUCTION
Human activity on the computer is an ad-hoc, messy process. Computer desktops are characterized by an array of running programs, e-mail, chat, to-dos and authored documents. To successfully manage their work, users often rely on windowing tools such as the Windows Taskbar [2] or Apple Exposé [1] to assist in switching and re-finding.

Task-based approaches offer promise for managing our growing workspaces. If computer desktops were meaningfully sorted into tasks, we might reduce cognitive overload by showing windows relevant to the user's current task and hiding others.

Whether explicitly communicated or implicitly learned, however, these task models present problems for the user. In the explicit case, the user may find it burdensome to explicitly classify windows into tasks. Even when the user is willing to undertake the activity or the system can classify automatically, the user may be hard-pressed to identify an objectively correct classification. For example, consider a user who visits Amazon.com to purchase books for his child's birthday but gets distracted by a related item and starts browsing other items on the site instead. Should we still call this activity the *buying a birthday book* task? Should we instead put it in a catch-all *distracted* task? Tasks also evolve constantly, and what may have been a correct classification an hour ago may be inappropriate now. For example, a user may write a document in the context of an *annual report* task, but the next day then refer to the document when making slides for a *boardroom presentation* task.

These issues bear directly on the design of window managers. In an evaluation of TaskTracer, users were often noncommittal when mapping windows to tasks: "…users are often not 100% sure themselves or may provide different answers in different contexts. Users are often able to tell the system what it is not, but not what it is." [10]

To address these problems, we extend a task model first explored in Rooms [4] and WindowScape [11] that allows windows to be identified with multiple tasks at the same time. Our research introduces *association*, a continuous measure of two windows' relatedness, into this model. We automate the association metric and its accompanying visualization. We believe such automation is critical, as users are typically disinclined to organize their personal information up front [3], and we hypothesize that they would be even less inclined to manually update this organization as tasks evolve and context changes.

In this paper, we apply our approach in a window visualization system called Taskposé. Underlying Taskposé is a set of algorithms for determining window importance and window association using both importance and window switch histories. Taskposé uses this association heuristic to lay out related windows near each other. We conducted a weeklong user study to evaluate Taskposé's accuracy and usefulness. Outstanding design needs include interface stability and evolution of our association algorithms.

UIST'08, October 19–22, 2008, Monterey, California, USA.

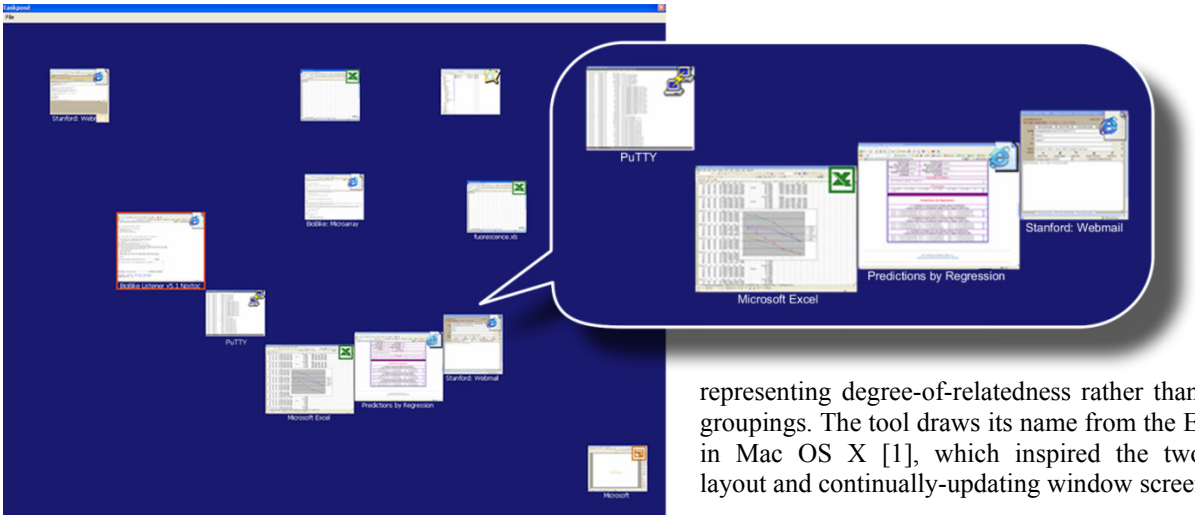

Figure 1. The Taskposé visualization arranges open windows in two dimensions when the visualization is called up. Windows automatically size relative to their importance, and closely-related windows appear together (inset).

RELATED WORK

Researchers have in general approached window manager design in two ways: by giving users manual control over the task organization of their windows, or by using machine learning to infer task structure. Manually controlled window managers rely on the user to define tasks. Such window managers include virtual desktops, Rooms [4], GroupBar [9], Scalable Fabric [8], and WindowScape [11]. Manual systems require significant investment from the user to manually organize windows into tasks in order to extract benefits. Predictive window managers, by contrast, assign a window to its most likely task based upon evidence of task creation and manipulation, such as window titles, activation history, and window content. When predictive window managers are accurate, they do not impose undue time and cognitive costs on the user. However, predictive systems that guess incorrectly impose repair costs on the user. Examples of predictive window managers include TaskTracer [10], SWISH [6], and UMEA [5].

Virtual desktops, WindowScape [11] and Rooms [4] are particularly relevant because they allow windows to selectively participate in multiple tasks. Rooms and virtual desktops allow users to place windows in multiple desktop contexts simultaneously. WindowScape builds on this approach by enabling implicit, non-binding task association: windows can exist in multiple tasks or in no task at all. Taskposé differs from these systems because it is predictive rather than manual and uses a degree-of-association visualization in place of explicit task groupings.

TASKPOSÉ

To accommodate complex relations between windows and users' tasks as well as the continually-updating nature of tasks, we have developed the Taskposé window switching visualization (Figure 1). Taskposé is a screen-filling visualization of the user's workspace in two dimensions, representing degree-of-relatedness rather than explicit task groupings. The tool draws its name from the Exposé feature in Mac OS X [1], which inspired the two-dimensional layout and continually-updating window screenshots.

Taskposé Layout

Taskposé represents open windows by thumbnails. The distance between thumbnails is based on the predicted association (degree of relatedness) between the windows. As the user exhibits behavior implying that windows are related to one another — in our system, by switching from one window to the other — these thumbnails move closer together on the Taskposé display (Figure 2). Groups of related windows will form as an emergent result of these window-to-window relationships. Windows that are independent of task, such as e-mail and music, tend to drift near the center of the visualization. A user may drag a window to another location or anchor a window with a right-click interaction, preventing it from moving until unanchored.

Taskposé has no explicit task groupings. Instead, it depends on the user's perceptual system to interpret the layout as a meaningful task organization, with proximity and empty space suggesting rather than imposing an organization. It is fundamental that the visualization can be understood in multiple ways, because a window may participate in multiple tasks. For example, a window related both to writing a report and to drafting a presentation should be interpretable as belonging to either group.

The Taskposé interface also visualizes the relative *importance* of windows, as estimated by our WindowRank algorithm. Window size in the visualization is directly correlated with the window's importance as estimated by our system (Figure 2), making important windows larger and thus easier to locate. Important windows also have more mass, so that they are relatively unlikely to move as the visualization updates. Keeping important windows relatively stable not only makes them easier to find, but also makes them more reliable anchors for task groupings.

Switching Windows

The Taskposé visualization may be opened in two ways: by double-clicking the Taskposé icon in the system tray, or by pressing Alt-` (backquote). This keyboard shortcut was chosen for its close physical similarity to the inveterate Alt-Tab key combination. When the Taskposé visualization

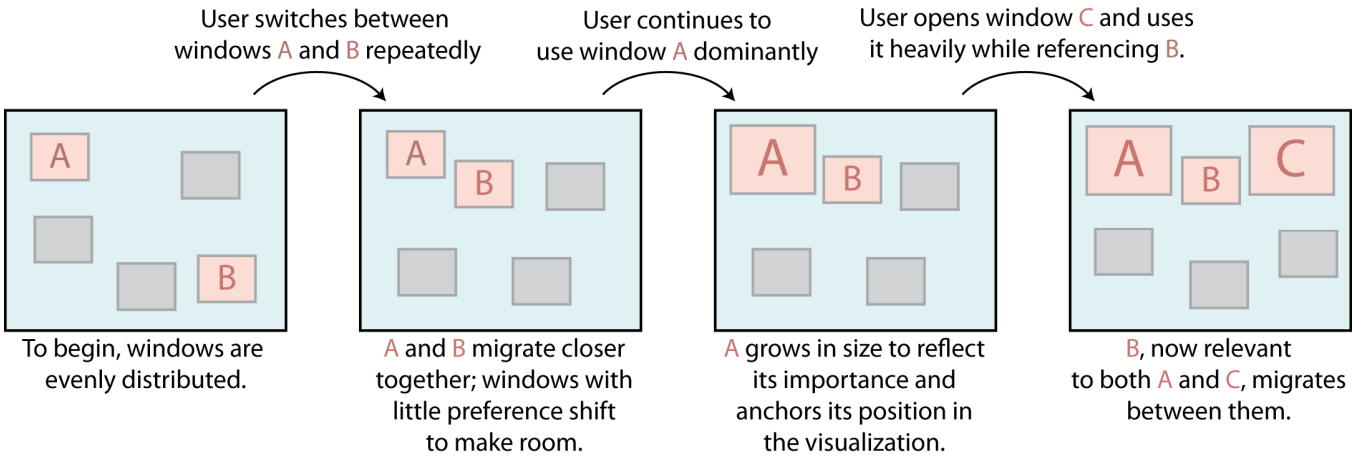

Figure 2. User interaction through the course of normal interaction with the computer will determine the Taskposé visualization.

appears, it overlays the contents of the user's screen and outlines the current window in red. To switch windows, the user clicks on the appropriate thumbnail. The Taskposé window then closes, and the operating system switches to the requested window. The visualization continues to update window locations while hidden.

IMPLEMENTATION

Three main algorithms underlie the Taskposé system: the WindowRank algorithm for determining window importance, the window relationship algorithm, and a spring-embedded graph layout algorithm.

The WindowRank algorithm takes as input a series of switches between windows in the operating system, and outputs a real number representing its determination of the importance of the window to the user's work. WindowRank builds on the approach popularized by Google's PageRank [7] by handling switches between windows similarly to PageRank's treatment of links between web pages. Where Switches*(A,B)* is the number of window switches from Window A to Window B, the iterative algorithm proceeds until convergence as follows, and is updated with each new switch:

$$\text{WindowRank}_1(A) = 1$$

$$\text{WindowRank}_{N+1}(A) = \sum_{X \neq A} \left(\text{WindowRank}_N(X) \times \frac{\text{Switches}(X,A)}{\sum_{Y \neq X} \text{Switches}(X,Y)} \right)$$

Thus, a window is deemed important if other (important) windows switch to it often. WindowRank is useful in the Taskposé context because information is collected without the user having to make any explicit assertions about relationships. Future work will integrate other features, such as window dwell time, into this heuristic.

Our window relationship algorithm aims to output a continuous metric of window association. Like previous work (e.g., [6]), we utilize window switches as a simple indicator of task relationship. In our algorithm, each window maintains a ratio of switches from itself to every other window. The algorithm then weights each window's switch ratio by a second ratio of the windows' WindowRanks:

$$\text{WeightedRatio}(A,B) = \frac{\text{Switches}(A,B)}{\sum_{X \neq A} \text{Switches}(A,X)} \times \frac{\text{WindowRank}(A)}{\text{WindowRank}(A) + \text{WindowRank}(B)}$$

$$\text{Association}(A,B) = \text{WeightedRatio}(A,B) + \text{WeightedRatio}(B,A)$$

Weighting by WindowRank allows more important windows to override other windows' preferences; this is useful in practice because newly opened windows may have very strong (but biased) views of their relationships based on a small number of window switches.

The window switch algorithm is a simple model of task relationship, and we propose it not as a solution to task tracking but rather as a proof of concept in support of an associative task relationship interface.

Given the *a posteriori* relationship computed between windows, a spring-based graph algorithm lays out the thumbnails. The result of this operation is that closely related windows are connected by short, stiff springs, and tend to cluster. Unrelated windows end up with long but loose springs. Windows move with mass proportional to their WindowRank, so important windows are mostly static.

The Taskposé prototype is implemented in Windows using C# and the .NET platform. It hooks into the Win32 API to listen to and publish window events, and to retrieve window icons, labels and screenshots.

EVALUATION

We performed a week-long longitudinal evaluation of the Taskposé prototype. Ten undergraduate students (five male, five female), all regular users of Windows XP, were recruited for this study. Taskposé was installed on their primary computer and the researcher demonstrated its use. For one week, participants used Taskposé for an hour a day as part of their everyday computer use, in addition to normal windowing tools such as the Taskbar and Alt-Tab. No specific task instructions were given, as we wanted to learn how the Taskposé visualization fared when given naturalistic data. After the week elapsed, researchers held a debriefing session and the participants answered a ques-

tionnaire about the experience. Users kept diaries, and system use was logged during the study.

Results

Surprisingly, we found that participants left the Taskposé software running in the background far longer than the required hour per day – the median over the week was 40.8 hours (min 10.3, max 195.4). Participants used Taskposé to switch windows a median of 156 times during the weeklong evaluation (min 19, max 237).

Users expressed an interest in continuing to use Taskposé, and generally found the relation and importance tracking to be useful in their everyday work (Figure 3). Eight of the ten participants found Taskposé to be most useful when the number of open windows outstripped space available on the Windows taskbar; in interviews, participants confirmed that the system's strengths were to be found in intense task-based work. However, none of these users had previous access to 2D window visualizations, so the result may be conflated with general approval of a 2D visualization approach. Users did not report the Taskposé visualization becoming cluttered or running out of space.

The design had several shortcomings. Visualization stability is an issue, though our evaluation revealed that anchoring and growing important windows in size often was a strong enough cue for most users. Users asked for additional control and customizability over the interface, such as being able to resize thumbnails and integrate drag/drop information into the association algorithm. Some participants stated a preference for a one-dimensional version of the program which could dock to the bottom of the screen just like the Windows taskbar.

Window Relationship Algorithm Shortcomings

Relationship tracking was rated as middling with a median rating of 4 on a 7-point Likert scale. Longitudinal use revealed several specific drawbacks. First, parent-child relationships were not accounted for: for example, users wanted chat windows to automatically group with each other and with the buddy list. Secondly, participants reported that when simultaneously working on multiple tasks, Taskposé would move the tasks close together; they had expected the distinct tasks to be spaced farther apart. As might be expected, tabbed web browsing was found to decrease the usefulness of the algorithm because the browser started associating with multiple groupings.

Future work can improve upon our model by integrating data sources other than window switches. Previous work often requires users to predefine tasks [5, 10] or produces hard boundaries between tasks [6]. However, much of this work can be adapted to the associative space, which is unsupervised and continuously-valued. Tools such as SWISH's window content PLSA approach or multidimensional scaling are strong candidates.

CONCLUSION

The Taskposé window switch visualization aims to speed window switching and aid workspace understanding by taking a perspective of task *association* rather than task *classification*. Our approach has three main ramifications: allowing windows to exist in multiple tasks, enabling windows to associate with tasks to varying degrees, and automatically updating visualizations as window relationships change. Implicitly, Taskposé dispenses with the notion of tasks altogether, as task relationships are emergent in the visualization rather than explicitly constructed. We believe the association approach provides many of the benefits of task organization without the cognitive or time tradeoffs associated with classification.

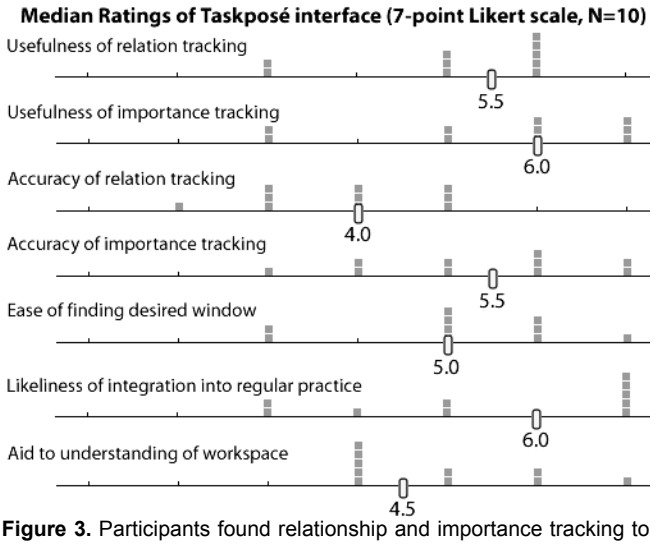

Figure 3. Participants found relationship and importance tracking to be useful to their work. The importance tracking algorithm received positive reactions. The response to relation tracking was middling.

REFERENCES

1. *Exposé*. Apple Computer, Inc. http://www.apple.com
2. *Windows Taskbar*. Microsoft. http://www.microsoft.com
3. Bernstein, M., Van Kleek, M., Karger, D.R., and schraefel, mc. Information Scraps: How and Why Information Eludes Our Personal Information Management Tools. To appear in *ACM Transactions on Information Systems*.
4. Henderson, D. A., and Card, S. Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics* 5(3), 211-243, 1986.
5. Kaptelinin, V. UMEA: Translating interaction histories into project contexts. *Proc. CHI 2003*: ACM Press. pp. 353-360, 2003.
6. Oliver, N., Smith, G., Thakkar, C., and Surendran, A. C. SWISH: Semantic analysis of window titles and switching history. *Proc. IUI 2006*: ACM Press. pp. 194-201, 2006.
7. Page, L., Brin, S., et al. The PageRank Citation Ranking: Bringing Order to the Web, Stanford Digital Libraries Working Paper, 1998.
8. Robertson, G., et al. Scalable Fabric: flexible task management. *Proc. CHI 2000*: ACM Press. pp. 494-501, 2000.
9. Smith, G., et al. *GroupBar: The TaskBar Evolved. Proc. OZCHI 2003*. 2003.
10. Stumpf, S., et al. Predicting user tasks: I know what you're doing! *Proc. AAAI 2005*: AIII Press. 2005.
11. Tashman, C. WindowScape: A Task Oriented Window Manager. *Proc. UIST 2006*: ACM Press. pp. 77-80, 2006.