Taskposé: Exploring Fluid Boundaries in a Task-Based Window Manager

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 managers assist users in navigating their computing workspaces by providing an organizational and access mechanism for their open windows. Window manager research has aimed to leverage users' tasks to organize the growing number of open windows in a useful manner. This research has assumed task classifications to be binary—a window is in a task, or not—and context-independent. We suggest that tasks' continual evolution can invalidate this approach and introduce *association* between artifacts as an alternative organizational scheme. Association relates windows to one another at varying degrees; task-relatedness is an emergent property of association. We describe Taskposé, our implementation of an associative window manager, 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 is characterized by complex patterns in both the physical and digital worlds. On physical desks, reams of paper and documents, books and writing tools evidence this activity [27]. Computer desktops are likewise characterized by an array of running programs, e-mail, chat, to-dos and authored documents [6, 8]. Research has found users to generally keep at least eight windows open on their desktop [21], and this number is sure to rise as screen space and memory become cheaper and more aspects of our lives go online.

In the physical world we use numerous means to organize challenging complexity: documents are sorted into piles, notes indicate reminders or items of priority, and perhaps we utilize a paper organizer or two [27]. Similar attempts to organize the computer desktop have been made, for example using complex file hierarchies and multicolored email flags, but overall the computer desktop has resisted such organization. Two prime examples, the Windows taskbar and the Mac OS X dock, siphon all open programs into a single pile at the bottom of the screen, offering little in terms of organizing principle other than time or program of origin.

We may expect to find a measure of control over this situation by considering the implicit tasks underlying users' activity. By "task" we mean a high-level goal towards which a person's actions are directed: writing an essay, paying bills, socializing, or researching camera prices are all examples. By minimizing the expected cost of finding a particular artifact [36], task-based approaches can be powerful tools for customizing one's workspace (see, *e.g.*, [9, 31]). In the physical world, the paraphernalia related to a given task may be sprawled out across the desk, while that of inactive tasks hovers in piles nearby. If the taskbar and desktop were meaningfully organized into tasks, human spatial memory and hierarchical thinking could likewise be leveraged to help us organize our computational lives.

Researchers have spent considerable effort on this proposition and introduced a variety of task-based systems which can automatically group windows into tasks [17, 23, 26, 32, 34] or give the user control over these groupings [5, 22, 38, 39, 41, 44]. This prior work has explored a variety of questions: What kind of organizational schemes do users employ most successfully? How can computer users communicate their tasks to the system? Do these need to be communicated explicitly, or can tasks be intuited from user actions? To what extent can users be troubled to organize such short-lived windows themselves, and to what extent should the computer help users automatically organize their work?

We have ourselves approached these questions in several steps. We begin by reporting on an observational study of how users typically organize their computer desktop during real work. Armed with a sense of how users create and interact with windows, and building on previous research, we propose an *association-based* task model wherein windows may identify with multiple tasks. We apply this approach in an automatic task and window organization system called Taskposé, wherein association offers both the motivation and mechanisms for tracking window relationships. Finally, we report on two evaluation studies of the Taskposé prototype.

RELATED WORK

Taskposé draws from related work in two broad categories: theoretical and observational research surrounding the nature of tasks in human-computer interaction, and prior window and task management systems. We address each in turn.

Tasks in Human-Computer Interaction

As the 'task' is not a well-defined concept, much prior work has been dedicated to exploring task boundaries and ramifications. The idea of task-based analysis of activities is a well-established theory in cognitive psychology (see, *e.g.* [15, 20, 30]), where it serves as a foundation for more recent work. Traditional HCI cognitive modeling (see, *e.g.* [11, 37]) has generally been concerned with micro-scale goals and objectives, making it more difficult to analyze the macro-sized tasks which are our concern: these can last hours and involve numerous interrelated goals. Activity Theory researchers have sought to address this problem by treating the computer interface as a medium through which users take action towards a goal [10, 24, 33]. Suchman [43] notes the importance of the larger task goal as an important mediator of action. Additionally, Winograd and Flores [45] caution us that task identification may become ad-hoc as pre-composed groupings break down during actual work.

Other research speaks to these theories by exploring worker tasks, interruptions and ad-hoc switching through ethnographic observation (*e.g.*, [4]) at differing levels of granularity. Observing entire task lifecycles, Czerwinski *et al.* [14] reported 53-minute task completion times on average and Bellotti *et al.* [8] saw a majority of tasks being completed within four hours. González and Mark [19] focused on a finer level of detail and found users spent only 2.5-minute stretches on electronic tools before task interruption and only a modestly higher 11 minutes at the more general level of the work sphere. Even our fine-grained interaction patterns are fragmented: Hutchings *et al.* [21] tracked window usage and discovered that participants spent a median amount of 3.77 seconds on a particular window before switching away; the mean was 20.9 seconds. Taken together, this work paints a startling picture of the prominence of task switching and interruption in our work.

Tasks and Window Managers

The commercial world contains dozens of window managers, from open-source customizations to commercial-grade software such as the Windows taskbar [2] and Apple OS X Exposé system [1]. We consider this generic window management work as it addresses underlying engineering and design issues.

Researchers have approached window manager design in two ways: either by building systems that can intuit users' task structure, or by giving up on such automation completely and instead granting users manual control over window and task organization. As a result task-based window managers fall into one of two categories: *agnostic* or *predictive*.

Agnostic window managers do not attempt to make any generalizations about users' tasks and rely on the users themselves to define the tasks as they work. The strength of this approach is that it does not make task classification mistakes. Agnostic window managers have been explored in many shapes and forms: Rooms [22], virtual desktops, the Task Gallery [39], GroupBar [41], the ABC Extension to Windows XP [5], WindowScape [44] and Scalable Fabric [38] are all examples. These types of systems offer their most significant return given an equally significant investment in manually organizing windows into tasks. Thus, we believe they are best suited to long-term tasks that operate in a static set of windows. Rather than exploring the relative merits of a new kind of agnostic interface, we have chosen to focus on the predictive space.

Predictive window managers utilize algorithms that assign a window to its mostly likely task. The clear advantage of predictive window managers is that, if they make correct decisions, they do not impose additional sorting time requirements on the user in order to extract some benefit. Of course, if the system makes incorrect decisions, users are generally worse off than if the computer had done nothing at all. This approach bases its decisions on (often indirect) evidence of task creation and manipulation, such as window titles, switch history and content evaluation.. Examples here include TaskTracer [17], Kimura [26], SWISH [34] and window-frequency algorithms [32]. UMEA [23], while not a window manager, follows a similar approach toward creating dynamically-updating project spaces.

FIELDWORK

The goal of our observational study was to inform our model of task creation and manipulation. To build on prior observational research, we focused on users' existing window usage patterns and adaptations around task work. We hoped to capture user mistakes and breakdowns, as they are often a useful starting place for future designs. By pulling out cross-user threads from this observation we hoped to provide evidence to guide our research.

Method

Our opening study was an in-situ observational visitation. We recruited subjects who came to use a university public computer cluster. Nine males and nine females participated; they were all undergraduate students, graduate students or visitors to the university. With permission, at some arbitrar-

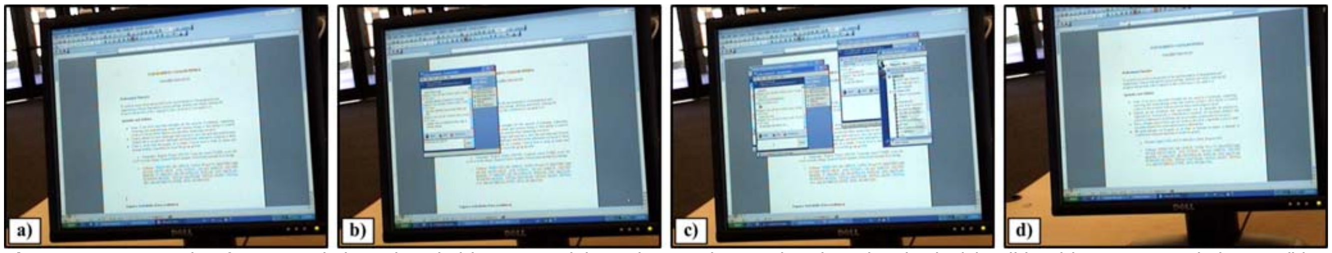

Figure 1 An example of opportunistic task switching: A participant in our observational study who is (a) editing his resume switches to (b) a chat window when his friend greets him, then is drawn to (c) several other unrelated chat windows before (d) returning to work.

ily chosen point during their ongoing work, we observed and video-taped the participants' actions, occasionally asking them questions about their ongoing activity. Each of the participants was observed and videotaped for a 5 to 15 minute period of normal computer use. At the conclusion of the study, the videotaped records were coded for window switches, number of windows open, and task relationships.

Results

Consistent with prior studies, we found users were generally engaged in multiple interleaved tasks. This multitasking often involved a central task and at least one peripheral task. One canonical example was that of a participant who wrote an email to her friends about a concert while referencing a web site and gathering e-mail addresses from an online student directory. Over the course of our study, we saw this central task encompass from one to seven simultaneous open windows, averaging around three windows.

Task switching is often opportunistic. Participants would sometimes leave a task with the intention of briefly pursuing another item only to launch into a completely new task—until eventually remembering to resume the previous work (Figure 1). Chat windows are emblematic of a class of windows that caused this phenomenon: generally unrelated to all other work, constantly referenced, but only active for short bursts of time. This class also included music players, sports tickers, and email clients.

We found participants' window switch frequency to be bimodal: most participants would settle in to a specific window and work without switching for a few minutes, then begin switching windows quickly and often for a period until finally settling again. This window thrashing activity [22] might signal a new task or sub-task, but just as likely the user was referencing the other windows, reorienting his or her workspace, trying to find a particular window or even taking a break (as in Figure 1). Users actively synthesizing information from multiple windows tended to exhibit similar behavior but had shorter dwell periods on the main window.

Discussion of the Observational Study

In our observations, users were generally aware of one task at any given time. As a result, 'orphaned' windows were often left open long after they were still in use, because users tended to forget about them and they do not make themselves apparent. The user's task space should therefore be considered random access, or perhaps center-surround [18], where the user is aware of the current task and only the sub- or super-tasks that are especially relevant to the project at hand.

Even though we preferentially chose subjects with multiple open windows, few of the participants exhibited complex multitasking behavior. We attribute this result to our locating the study in a public computer cluster. Public computers are generally used for short periods of time and for single purposes such as checking email; this environment discouraged multitasking. As evidence in support of this explanation, participants who had brought their own laptops to use in the cluster exhibited a far greater number of multitasking behaviors than participants using public computers.

Our account of desktop multi-tasking might be summarized as follows:

- Users generally work on a single main task at a time, often spanning multiple windows.
- Task switching does not often happen between main tasks—users tend to work in coherent bursts. However, short switches between the current main task and background items such as chat, music or email are not uncommon.
- New tasks or subtasks are spun off opportunistically. Old threads are often left behind if some new work becomes high-priority, or if the trail leading back to it becomes too long. This results in windows sometimes switching task association quickly, and sometimes migrating between associations over a long period of time.

FROM CLASSIFICATION TO ASSOCIATION

Previous research explicitly assigns windows to a specific task group—a window is either part of one task, or it is part of another. Our work's contribution lies in incorporating the claim that tasks are "fuzzy" and have continuously changing relationships with their contents. We build on a small but growing set of literature that indicates that task *classification*, an approach in which work artifacts are placed strictly in one task, is an improper match for users' mental models.

One interesting result arises from the evaluation of the machine-learning techniques applied to TaskTracer [42]. In their study, the authors asked users to evaluate whether

TaskTracer had made a correct task classification prediction based on their activity. These researchers found that users were often unsure which task a window should be allied with: "…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" [42]. In an evaluation of the Activity-Based Computing extension to Windows XP, a user likewise mused: "The worst thing? Well [...] if you have to put everything into activities, then you need to constantly consider 'where does this one belong'" [5].

It is important to note here that because users are aware of their own higher-level goals, they should in theory know the classification of every window. However, users' difficulty with the sorting operation suggests this awareness may not always be present. Because both of these systems allow for arbitrary naming of tasks, the classification systems in use cannot be causing this difficulty. Instead, we believe that the single task classification model does not always map well onto users' mental models of their work. This situation is essentially the problem of asking pilers—who often delay sorting of artifacts—to live in a world where filing is the only option [27]. It is worsened by requisite mental upkeep in the form of continuous filing of new windows.

To give an example of the problem at hand, imagine a fictitious user who is beginning a new task of buying a book. The user logs on to an online shopping web site in order to purchase the book, then is distracted by a related item and begins browsing related works. From a task perspective, is the user still buying the book? Is the user really *not* buying the book? This situation is an example of a gray area with regards to clean mapping. Or, when a user writes a document under a yearly report task, but later refers to the document when generating a set of slides for a boardroom presentation, should the paper be part of the business report task, or the boardroom presentation task? Or both? Here, we see a situation where an artifact's task classification changes with a context switch.

Bowker and Star [11] address this concern as part of a larger argument on the consequences of classification. They define classification as "a spatial, temporal, or spatio-temporal segmentation of the world" characterized by (1) consistent decision principles, (2) mutually exclusive categories and (3) the union of the categories encompassing all possibilities. The authors point to examples of our "muddled folk classification":

> *A quick scan of one of the author's desktops reveals eight residual categories represented in the various folders of email and papers: "fun," "take back to office," "remember to look up," "misc.," "misc. correspondence," general web information," "teaching stuff to do," and "to do." We doubt if this is an unusual degree of disarray or an overly prolific use of the "none of the above" category so common to standardized tests and surveys.* [11]

The work above supports a hypothesis that the relationship between tasks and actions is not one-to-one and suggests that it is preferable to build systems which handle classification more flexibly. For example, research on the Piles [28] and Placeless Documents [16] projects have supported less strictly defined organizational schemata. More recently, this general phenomenon has also exhibited itself on the web via the rise of folksonomies, which espouse greater flexibility than traditional filing [40]. Our work also explores this gray area of task classification.

A New Task Model: Association

Based on the foregoing observations and previous research, we believe that the following two kinds of situations are a common use case that must be considered in the design of task-oriented windows managers:

1. Users' task classifications come in many shades, which strict groupings cannot support, and
2. Strengths of association between artifacts may change over usage time, or immediately if the context switches.

With the exception of WindowScape [44], research into task-based windows managers has assumed that windows are cleanly mapped into a specific task and that windows are statically part of one task. Following computer science terminology as well as Bowker and Star, we term such window managers as performing *classification*. Classification is defined as treating task decisions as a binary yes-or-no problem: is this window part of this task, or isn't it?

By way of contrast, we define *association* as allowing artifacts to identify with tasks at varying degrees. A window can be strongly associated with a single task, weakly associated with several tasks or associated with none at all. Our goal is to design a window manager that incorporates association in a useful and user-friendly way.

TASKPOSÉ

In order to support complex desktop activity, we have developed an associative window manager called Taskposé wherein window icons appear in a two-dimensional space (Figure 2). It draws its name from Apple Exposé [1], which inspired the system's two-dimensional layout and use of continually-updating window screenshots.

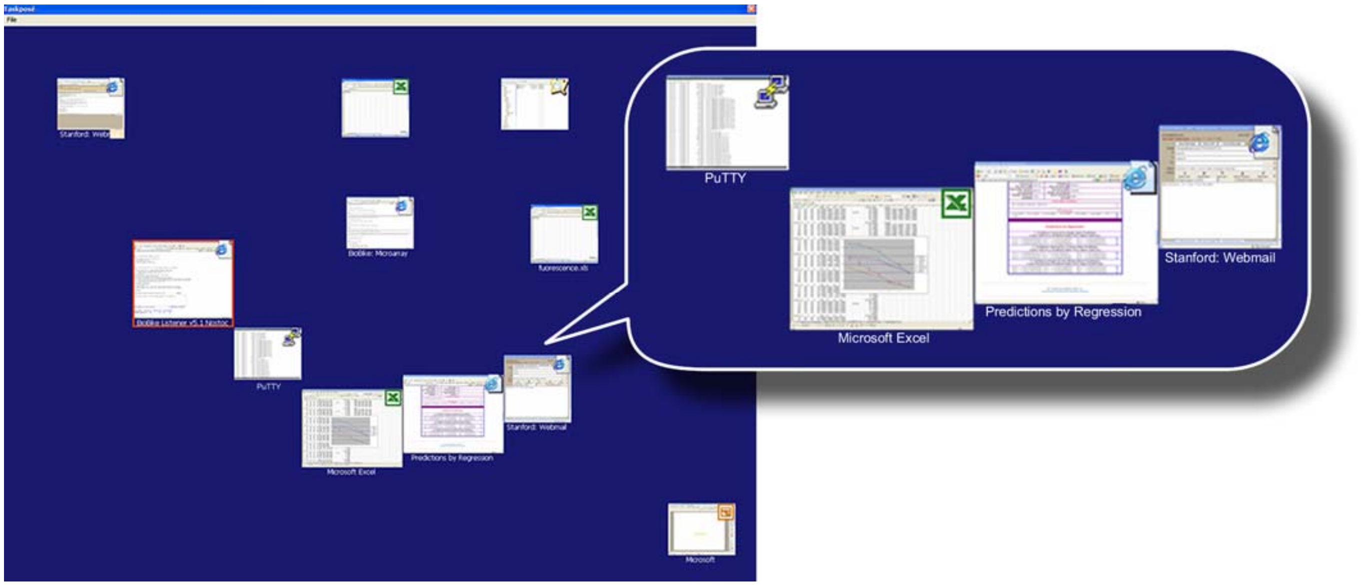

Figure 2 The Taskposé visualization arranges open windows in two dimensions when the visualization is called up. Windows automatically size relative to their importance. *Inset:* closely-related windows appear together in the visualization.

Taskposé represents open windows by their thumbnails, which continuously update while the visualization is open. The distance between these thumbnails is tied to the predicted semantic (i.e., task-based) relation between windows. Taskposé, in fact, has no task groupings at all. Rather, as users exhibit behavior implying that windows are related to one another, the icons move closer together on the task manager display. The user's Gestalt organizational capacities permit him or her to interpret this layout as a meaningful task organization—the spacing 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 to writing a paper and to drafting a presentation should be easily interpretable as belonging to either group.

The decision to replace actual task groupings with visible degrees of association was motivated by our hypothesis that task classifications are not binary decisions to be made. Likewise, the continuously-updating nature of the visualization, as well as the ability of windows to remain between two clusters, supports the idea that tasks are context-dependent and may change over time. We made a design decision to visualize the user's workspace in two dimensions because we felt it represented interdependencies better than window icons arranged in a one-dimensional line such as in the Windows taskbar or OS X dock.

Interaction with Taskposé

A few simple rules guide Taskposé's user interface. First, distance between window pairs is determined by how related Taskposé believes them to be. Tightly related windows will thus move right next to each other (Figure 2 inset), and unrelated ones remain apart. Groups of any number of windows may form in this manner. Windows related to several disjoint groups will appear between those groups in the visualization. A user may anchor a window via a right-click interaction, preventing it from moving unless unanchored. The user can also move a window to another part of the visualization via a drag interaction if he or she wishes. Taskposé does not currently interpret drag-and-drop location as new relationship information.

Important windows inform other aspects of the Taskposé interface. Most critically, window size in the visualization is directly correlated with the window's importance, as estimated by Taskposé (Figure 2). One major design concern was that windows would move from remembered locations while the user wasn't looking, and thus he or she would have difficulty relocating windows. Thus, in Taskposé, important windows have more "mass": they move less, if at all, as the visualization updates. This weighting of important windows trades on an assumption that important windows are typically the ones that users will want to find, and will be the most disruptive if they unexpectedly shift.

Switching Windows

A complete interaction with Taskposé takes only a few seconds. The Taskposé visualization may be brought up in one of two ways:

1. Double-clicking the Taskposé icon in the system tray.
2. Holding the Alt key and pressing the ` (Accent Grave) key. This interaction was chosen for its close physical similarity to the inveterate Alt-Tab key combination—most users preferred this method for its speed and ease.

When the Taskposé visualization appears, it overlays the contents of the user's screen and outlines the current window in red to help orient. To switch windows, the user clicks on the appropriate thumbnail. When the user clicks on a window, Taskposé hides and the operating system switches to the requested window. If the user decides not to switch windows, he or she can hide the visualization by repeating either of the show mechanisms above.

IMPLEMENTATION AND ALGORITHMS

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 as well as to retrieve window icons, labels and screenshots. Three main algorithms underlie the Taskposé system: the WindowRank algorithm for determining window importance, the window relationship algorithm, and the graph layout algorithm.

The WindowRank 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. Other algorithms have attempted to utilize window switching to determine window relevance with reasonable success [32, 34], but to our knowledge none have attempted to do so to describe window importance. We later use this importance metric to inform our relatedness algorithm.

WindowRank builds on the approach popularized by Google's PageRank [35]. PageRank treats the Internet as a series of nodes on a graph, and links between pages as edges on that graph. A web page's PageRank is determined by the accumulated PageRank of web pages linking to it. WindowRank treats windows as the nodes in the graph and user window switches as edges. So, each time a user switches from Window A to Window B, WindowRank treats the action as Window A voting for Window B and adds a proportion of its own rank to the destination.

WindowRank is useful in the Taskposé context for several reasons. First, information is collected without the user having to make any explicit assertions about relationships. The algorithm runs quietly every time a new user action occurs: for example, a window switch, open, or close. Because the number of graph nodes is relatively small, the algorithm in practice runs quite quickly and does not become a performance issue. Second, as we shall see, knowing which windows are important to the user's work plays a critical role in differentially weighting windows' opinions about what is related to what.

The Window Relationship Algorithm

The most important utilization of WindowRank appears in the window relationship algorithm. It was our goal to fashion an algorithm that would output associative relatedness over a continuous region, which we could then incorporate into our visualization. Our algorithm takes as input window switches and window ranks and then outputs a real number in (0, 1) representing a weighted judgment of the strength of the relationship between the two windows. 0 corresponds to totally unrelated, and 1 corresponds to extremely closely related.

There are many classes of algorithms which might fill this role. We have chosen one which, while fairly simplistic, serves well as a proof-of-concept algorithm for Taskposé. Our algorithm is similar to other window switch relationship algorithms, but is unique in its incorporation of window importance. We believe that this consideration to be useful in improving the accuracy of such algorithms.

WindowRank and importance are necessary here because Window A and Window B may have different opinions about how closely related they are to each other. For an explanation, consider a naïve algorithm which treats both Window A and Window B as equals in the decision. If Window A is an important window, it will likely have switched to and from different windows many times. So, its vote for B will be relatively small, but likely accurate, as the user has not evinced much behavior indicating a strong relationship between the windows. On the other hand, if Window B is unimportant and thus sees fewer switches, each switch to A will greatly influence B's opinion of its relationship to A. Here, by averaging A and B's guesses, the naïve algorithm will return an over-inflated estimate of the windows' relationship.

WindowRank reduces this problem by allowing important windows to override unimportant windows' over-inflated claims. The algorithm in use weights each vote by the ratio of its rank to the two windows' ranks summed:

$$\begin{aligned} \mathit{Association(A,B)} = {} & \frac{\mathit{Switches(A,\, B)}}{\sum_X \mathit{Switches(A,\, X)}} \cdot \frac{\mathit{WindowRank(A)}}{\mathit{WindowRank(A)} + \mathit{WindowRank(B)}} \; + \\ & \frac{\mathit{Switches(B,\, A)}}{\sum_X \mathit{Switches(B,\, X)}} \cdot \frac{\mathit{WindowRank(B)}}{\mathit{WindowRank(A)} + \mathit{WindowRank(B)}} \end{aligned}$$

This returned value is between 0 and 1, and is used by the Taskposé visualization to display window relationships.

Graph Visualization and Updating

Given the a posteriori relationship computed between windows, a spring-based graph algorithm [7] (also known as a mass-spring model) lays out the icons. In a spring-embedded graph layout, a simulated spring is attached between every two nodes in the graph, and spring physics continually adjust node locations. For example, two nodes connected by a short, stiff spring will stay near each other. We decided upon a spring-embedded graph because of its aesthetic layout characteristics and its ability to map continuous values from our relatedness algorithm directly onto visual distances.

The output of the window relationship algorithm is linearly mapped onto both the spring length and stiffness for each pair of windows. 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; this is desired so that there is some flexibility in the windows' relative placement.

During each program cycle, every window is moved by an amount proportional to the overall force acting on it by all the springs connected to it. This proportion is determined by each window's WindowRank. That is, important windows which have above average WindowRank move less, and unimportant windows (with less WindowRank) move proportionately more. This allows the graph to smoothly update without disrupting the positions of important windows.

EVALUATION METHOD

To evaluate Taskposé, we wished to test the following hypotheses:

H1 Taskposé's approach of associating rather than classifying windows maps well onto users' mental models of their work.

H2 Taskposé successfully scales to situations with many windows open.

H3 Taskposé's window importance and relationship tracking algorithms are powerful enough to avoid negatively interfering with users' evaluations of H1 and H2.

We conducted two studies of Taskposé, encompassing a pair of study designs: a first-use study and a longitudinal study. These studies elicited four different types of data collection: free form interview, self-reported questionnaire, videotaped observation and computer-generated usage log. Participants were not told of Taskposé's underlying algorithms until after each study was completed.

First-Use Study

We chose a first-use study for its power to rapidly elicit usability problems. Ten undergraduate students at our university (six male, four female) were recruited to take part in the forty-five minute study. Sessions were held on the participants' own computers or on the researcher's laptop. Personal computers' resolution varied, though the laptop was always set at 1280x1024 pixels.

First, the researcher gave a tour of the interface. Then, the participant was presented with a task to compile information from several Internet web sites. This task was inspired by the multitasking activities we observed in our fieldwork study. Specifically, participants were asked to find specified information about the Political Science programs at four major universities. This information was to be compiled into a separate document for each Political Science program. Participants were given 20 minutes to complete the task.

The task required numerous window switches and caused a great deal of window thrashing [22]. We encouraged participants to use Taskposé when switching windows, but they were not required to do so. Participants followed a 'think-aloud' protocol during completion of the task: this vocalization of participants' inner thoughts and confusion clarified the user's mental model of the program to the researcher at moments of breakdown. Further, the researcher observed and videotaped participants' interactions with the system.

As the purpose of this short study was mainly to elicit usability problems and iterate on Taskposé's design, we did not attempt to collect quantitative data. The results of this study were incorporated in the next version of Taskposé and led into the longer, more substantive longitudinal evaluation.

Longitudinal Evaluation

Due to the background nature of window managers and the wide variety of taskbar use styles, we felt that allowing Taskposé to be used in conjunction with everyday work practices and over an extended period would produce a more compelling measure of its success or failure. The main strength of a longitudinal approach lies in testing the sustainability and scalability of our design; its main drawback is that allowing users to use the software on their own time precluded a researcher from observing the interaction.

Ten undergraduate students (five male, five female) were recruited for this study. Taskposé was installed on their main computers, and the researcher demonstrated its use. For one week, participants used Taskposé in the course of their everyday computer work for an hour a day. Participants who wished were allowed to use Taskposé more than the required seven hours. No specific task instructions were given, as we were interested in as naturalistic an experience as possible. Each participant was given a logbook in which to record reactions to the software during use sessions, which would be reviewed by the researcher at the culmination of the study.

After the week elapsed, researchers held a debriefing session and the participants answered a questionnaire about the experience. With regards to the interface itself, the questionnaire contained a series of Likert scale questions designed to measure the accuracy and usefulness of window importance and relationship tracking, and ease of finding the desired window. On a broader level, we inquired after the system's contribution to users' understanding of their workspace, enjoyment, and their likelihood of integrating a "perfect" version of Taskposé into their everyday work habits. Additionally, the software kept automatic usage logs so the researcher could analyze data such as typical number of open windows and program usage frequency.

RESULTS

This section highlights several outcomes of the longitudinal use study. We organize the results around analysis of our hypotheses.

Association over Classification

Users generally expressed an interest in continuing to use Taskposé in their everyday computer work. On a 7 point Likert scale, users responded with a median score of 6 when asked how likely they would be to integrate a perfect version of the system into their regular work practice (Figure 3). Enjoyment was likewise rated highly, with a median response of 5.5.

Participants' Taskposé use time (hrs)	*Total Taskposé window switches*	*Taskposé usage rate (switches/hr)*
195.4	196	1.00
118.2	237	2.01
64.7	156	2.41
57.9	181	3.12
40.8	48	1.18
20.8	75	3.61
13.9	21	1.511
12.1	19	1.57
10.3	161	15.58

Table 1 Users kept Taskposé open far beyond the required seven hours, leading to an artificially low recorded usage rates.

Users' willingness to integrate an idealized system into their regular workflow, especially given the limits they had experienced with the window relationship algorithms, suggests that Taskposé's grouping approach did in fact map well to users' mental models of their work (H1). In interviews, participants generally confirmed that they were in favor of the visualization strategy, especially during intense task-based work. Often users asked for additional control and customizability over the interface, such as being able to resize windows and integrate drag/drop information into relationship knowledge. This suggests that they wished to further incorporate individual working styles in their use of the software. No users mentioned that strict task groupings would have been preferred, or suggested using them as an interface alternative at all.

Scaling

When asked for classes of situations when Taskposé was or wasn't useful, eight of the ten participants responded that Taskposé was much more useful when the number of open windows outstripped the space available on the Windows taskbar. Most participants preferred Taskposé to the taskbar and alt-tab in this kind of situation. This feedback lends strong support to H2. As expected, respondents commented that using Taskposé was a burden when the taskbar was still a viable option, or when switching back and forth between two windows repeatedly made it simpler to use the alt-tab key combination.

Window Relationships and Importance

Users found Taskposé's relationship and importance tracking to be quite useful, though they criticized the underlying algorithms' accuracy. Figure 4 summarizes users' evaluations of the importance and relationship tracking algorithms in Taskposé. Users' comments reflected this support for the functionality; for example: "The best part was this unique way of grouping windows that were most important, which Taskposé did accurately and which I'd never seen before." In general, the positive reactions to the functionality supports hypothesis H3 and reinforces our belief that association is a positive task management model (H1).

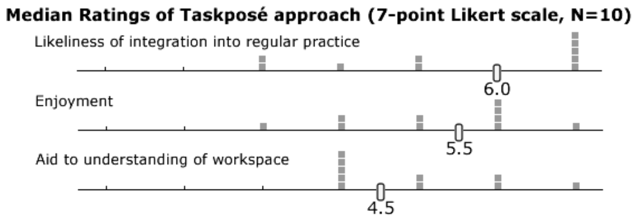

Figure 3 Participants were generally enthusiastic about incorporating Taskposé model into their everyday computing. Workspace understanding was generally unaffected.

Freeform comments revealed a few classes of problems with the relationship algorithm. First, parent program relationships were not accounted for in the Taskposé model: for example, users wished chat windows to automatically group with each other and with the buddy list. Secondly, participants reported that when working on multiple tasks, they found Taskposé would occasionally move the tasks close together as a result of their switching between the tasks; they had expected the distinct tasks to be spaced farther apart. Finally, as expected, tabbed Internet browsing was found to decrease the usefulness of the algorithm because the browser started associating with multiple groupings. Because tabbed browsing is attempting to solve the same problem as Taskposé—controlling an overabundance of windows—it is not surprising that their orthogonal approaches might interfere.

Logging Results Reveal Extended Use

Nine of the ten participants returned usable activity logs (one user's logs were corrupted). We attempted to analyze this log data to gather more quantitative usage information. Specifically, we intended to measure how often participants switched programs using Taskposé, and compare that to the number of times they switched using other means such as the taskbar, alt-tab, or by simply clicking on the window.

Surprisingly, we found that participants left the Taskposé software running in the background for extended periods of time—one user logged eight days straight—and often used it continuously. Thus, we were left with no indication of when they began their official usage hour each day. This behavior might have been expected: Taskposé does not track window relationships when not running, so leaving the software on in the background guaranteed it would be useful at the beginning of the usage hour. Though we are left with artificially low Taskposé switch rates as a result of this behavior (Table 1), it is an encouraging suggestion as

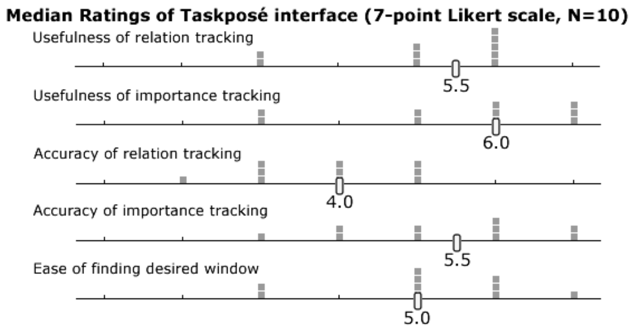

Figure 4 Participants found relationship and importance tracking to be useful to their work. The importance tracking algorithm received positive reactions, though the response to relation tracking was middling.

to users' ability to incorporate the Taskposé model successfully into their workflow.

Design Improvements

The study also elicited a set of design suggestions for the software. In this section we review several of the most promising suggestions based on the long-term evaluation.

Users tended to either use the anchoring functionality very lightly, if at all. Mainly users credited this to a lack of discoverability, as it was hidden in a right-click menu beneath each window icon. A simple design solution would be to use a pushpin metaphor [3] to make the interaction more visible. In our envisioned prototype, a specific corner of each window would be treated as a hotspot to allow one-click anchoring at the current location.

A second line of design feedback suggested that we scale the window thumbnails to fill the visualization at all times. Under this change, when two windows are open, they would appear large to fill the Taskposé display, but shrink to fit a third window when one is opened. This leads to problems with the consistency of window position, but is certainly a useful direction to consider.

The next major step in interaction for Taskposé is to allow users to specify strengths of association if they wish. For example, a user might position a window near a specific grouping to indicate that the window is strongly associated with its new neighbors. In effect, Taskposé would remember manually-specified associations. Such a system, if designed well, could lessen the need for a perfectly accurate association prediction algorithm.

CONCLUSIONS AND FUTURE WORK

In this paper, we introduce Taskposé, a window manager that embodies task association rather than classification as its core premise. In our user studies of the system, we found that the approach mapped well onto users' mental models of their work. Taskposé's current limitations lie primarily in the accuracy of its window relationship tracking. Several lines of predictive task management research have proposed other methods; however, the mathematical algorithms underlying these machine learning solutions are generally able to only make classification decisions or probabilistic guesses rather than generate the continuous strength-of-association numbers that Taskposé requires, so adaptation of these algorithms would be necessary to support the Taskposé model. Regardless, doing so would probably yield the single most significant increase in Taskposé's usefulness. Pursuing other data tracking techniques, such as concentrating on window dwell time, parent/child relationships, as well as algorithms such as Bayesian updating and multidimensional scaling [13] could also prove fruitful. We are also interested in examining richer data to inform our algorithms; tracking eye movements and visual attention [25] is an especially good candidate.

With regards to the interface itself, Taskposé's extended use study brought to light several dimensions of interactivity within its visual grouping paradigm. While we experimented with a linear mapping between associative relationship and visual distance, other sorts of mappings (such as logarithmic) might lead to stronger visibility of in-group and out-group status. Additionally, a few participants stated that they would have preferred a one-dimensional version of the program which could dock to the bottom of the screen just like the Windows taskbar, thus obviating the need to call up the visualization—the ideal design for such a system is certainly a direction for future research.

And here we finally return to the big picture question: how to organize our complex world. It is all too tempting to treat piles of post-it notes and an unorganized computer desktop as simply in need of filing. Yet we have seen that in some circumstances it is undesirable, if not impossible, to accurately file and re-file our life. We put forth an always-changing landscape as a richer and more promising notion.

ACKNOWLEDGEMENTS

This work was completed at Stanford University. The authors would like to thank the great number of researchers who lent their support, including Scott Klemmer, Todd Davies, Erica Robles, Victoria Bellotti, Jim Thornton, Ed Chi, Duke Hutchings, Simone Stumpf, Jon Herlocker, Greg Smith, Pat Langley, and Stephen Voida. Early versions of Taskposé were supported by code from Jon Herlocker and Duke Hutchings. Taskposé software is freely available for use by contacting the authors.

REFERENCES

1. *Exposé*. Apple Computer, Inc. http://www.apple.com
2. *Windows Taskbar*. Microsoft. http://www.microsoft.com
3. *OPEN LOOK*. Sun Microsystems, Inc. http://www.sun.com
4. Bannon, L., Cypher, A., Greenspan, S., and Monty, M. L. (1983). Evaluation and analysis of users' activity organization. *Proc. CHI 1983*: ACM Press. pp. 54-57, 1983.
5. Bardram, J., Bunde-Pedersen, J., and Soegaard, M. Support for Activity Based Computing in a Personal Computing Operating System. *Proc. CHI 2006*: ACM Press. pp. 211-220, 2006.
6. Barreau, D. and Nardi, B. A. Finding and reminding: file organization from the desktop. *SIGCHI Bull.* 27(3), 39-43, 1995.
7. Battista, G.D., Eades, P., Tamassia, R. and Tollis, I.G. *Graph drawing algorithms for the visualization of graphs.* New Jersey: Prentice Hall, 1999.
8. Bellotti, V., Dalal, B., Good, N., Flynn, P., Bobrow, D. G., and Ducheneaut, N. What a to-do: Studies of task management towards the design of a personal task list manager. *Proc. CHI 2004*: ACM Press. pp. 735-742, 2004.
9. Bellotti, V., Ducheneaut, N., Howard, M., and Smith, I. Taking email to task: the design and evaluation of a task management centered email tool. *Proc. CHI 2003*: ACM Press. pp. 345-352, 2003.
10. Bødker, S. *Through the interface: A human activity approach to user interface design*. Hillsdale, NJ: L. Erlbaum, 1991.
11. Bowker, G. C. and Star, S. L. *Sorting things out: Classification and its consequences.* Cambridge, MA: MIT Press, 1999.

12. Card, S. K., Newell, A., and Moran, T. P. *The psychology of human-computer interaction*. Hillsdale, NJ: L. Erlbaum, 1983.
13. Cox, T. F. and Cox, M. A. *Multidimensional scaling*. London: Chapman & Hall, 2000.
14. Czerwinski, M., Horvitz, E., and Wilhite, S. A diary study of task switching and interruptions. *Proc. CHI 2004*: ACM Press. pp. 175-182, 2004.
15. Damos, D., Ed. *Multiple-Task Performance*. London: Taylor & Francis, 1991.
16. Dourish, P., Edwards, W. K., LaMarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry, D. B., and Thornton, J. Extending document management systems with user-specific active properties. *ACM Trans. Inf. Systems* 18(2), 140-170, 2000.
17. Dragunov, A. N., Dietterich, T. G., Johnsrude, K., McLaughlin, M., Li, L., and Herlocker, J. L. TaskTracer: A desktop environment to support multi-tasking knowledge workers. *Proc. IUI 2005*: ACM Press. pp. 75-82, 2005.
18. Furnas, G. W. Generalized fisheye views. *Proc. CHI 1986*: ACM Press. pp. 16-23, 1986.
19. González, V. M. and Mark, G. “Constant, constant, multi-tasking craziness": managing multiple working spheres. *Proc. CHI 2004*: ACM Press. pp.113-120, 2004.
20. Hoc, J.M. *Cognitive psychology of planning.* San Diego, CA: Academic Press Professional, 1988.
21. Hutchings, D. R., Smith, G., Meyers, B., Czerwinski, M., and Robertson, G. Display space usage and window management operation comparisons between single monitor and multiple monitor users. *Proc. AVI 2004*: ACM Press. pp. 32-39, 2004.
22. Jr., D. A. H., and Card, S. 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.
23. Kaptelinin, V. UMEA: Translating interaction histories into project contexts. *Proc. CHI 2003*: ACM Press. pp. 353-360, 2003.
24. Kaptilenin, V., and Nardi, B. A. *Activity theory: Basic concepts and applications.* http://www.sigchi.org/chi97/proceedings/tutorial/bn.htm. 1997.
25. Kumar, M. GUIDe: Gaze-enhanced User Interface Design. http://hci.stanford.edu/research/GUIDe/
26. MacIntyre, B., Mynatt, E. D., Voida, S., Hansen, K. M., Tullio, J., and Corso, G. M. Support for multitasking and background awareness using interactive peripheral displays. *Proc. UIST 2001*: ACM Press. pp. 41-50, 2001.
27. Malone, T. W. How do people organize their desks?: Implications for the design of office information systems. *ACM Trans. on Information Systems*, 1(1), 99-112, 1983.
28. Mander, R., Salomon, G., and Wong, Y. Y. A “pile” metaphor for supporting casual organization of information. *Proc. CHI 1992*: ACM Press. pp. 627-634, 1992.
29. Mark, G., Gonzalez, V. M., and Harris, J. No task left behind?: examining the nature of fragmented work. *Proc. CHI 2005*: ACM Press. pp. 321-330, 2005.
30. Miller, G., Galanter, E., and Pribram, K. *Plans and the structure of behavior.* New York: Holt, Reinhart and Winston, 1960.
31. Moran, T.P. Activity: Analysis, design and management. *Proc. Symposium on the Foundations of Interaction Design*. 2003.
32. Nair, R., Voida, S., and Mynatt, E. D. Frequency-based detection of task switches. *Proc. HCI 2005*: British Computer Society. pp. 94-99, 2005.
33. Nardi, B. A. *Context and consciousness: Activity theory and human-computer interaction.* Cambridge, MA: MIT Press, 1996.
34. 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.
35. Page, L., Brin, S., Motwani, R., and Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web, Stanford Digital Libraries Working Paper, 1998.
36. Pirolli, P. and Card, S.K. Information foraging. *Psychological Review* 106(4), 643-675, 1999.
37. Pirolli, P., Cognitive engineering models and cognitive architectures in human-computer interaction, in *Handbook of applied cognition*, F.T. Durso, Ed. West Sussex, England: John Wiley & Sons. pp. 443-477, 1999.
38. Robertson, G., Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D. R., Meyers, B., Robbins, D., and Smith, G. Scalable fabric: Flexible task management. *Proc. AVI 2004*: ACM Press. pp. 85-89, 2004.
39. Robertson, G., van Dantzich, M., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D., and Gorokhovsky, V. The task gallery: A 3D window manager. *Proc. CHI 2000*: ACM Press. pp. 494-501, 2000.
40. Smith, G. Atomiq: Folksonomy: social classification. http://atomiq.org/archives/2004/08/folksonomy_social_classification.html, Aug. 2004.
41. Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., Robbins, D., and Andrews, D. GroupBar: The TaskBar evolved. *Proc. OZCHI 2003*: ACM Press. pp. 34-43, 2003.
42. Stumpf, S., Bao, X., Dragunov, A., Dietterich, T. G., Herlocker, J., Johnsrude, K., Li, L., and Shen, J.Q. Predicting user tasks: I know what you're doing! *Proc. AAAI 2005*: AIII Press. 2005.
43. Suchman, L. A. *Plans and situated actions: The problem of human-machine communication.* Cambridge University Press, 1987.
44. Tashman, C. WindowScape: A Task Oriented Window Manager. *Proc. UIST 2006*: ACM Press. 2006.
45. Winograd, T., and Flores, F. *Understanding computers and cognition: A new foundation for design.* Norwood, NJ: Ablex Publishing Corporation, 1986.